UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

MACHINE LEARNING APPLICATIONS FOR GEOSCIENCE PROBLEMS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

RAFAEL AUGUSTO PIRES DE LIMA
Norman, Oklahoma
2019

MACHINE LEARNING APPLICATIONS FOR GEOSCIENCE PROBLEMS

A THESIS APPROVED FOR THE
SCHOOL OF GEOSCIENCES

BY THE COMMITTEE CONSISTING OF

Acknowledgements

I want first to thank my mother and father, Maria Alaide Pires de Lima, and Nelson Augusto de Lima, as well as my siblings Silvio, Silvia, Marcos, Ana, my uncle Jose, and my extended family for their help and support throughout this project. I would like to thank my committee members: Dr. Kurt Marfurt, Dr. Shankar Mitra, Dr. Michael Behm, Dr. Zulfiquar Reza, and Dr. Thomas Neeson for their support and encouragement. I am very thankful for my supervisor Dr. Kurt Marfurt, who gave me the opportunity to learn more about geophysics and how to conduct research. I am extremely grateful for his support and guidance. To Rebecca Fay, thank you so much for helping me with all the many forms I belated submitted and for helping draft so many letters. I acknowledge CNPq-Brazil (grant 203589/2014-9) for graduate sponsorship and CPRM-Brazil for granting my leave of absence.

I would like to thank all the friends I made during the four years I spent at the University of Oklahoma for all the help they provided when I was working towards two masters, a graduate certificate, a doctorate, and a graduate research assistantship in the Los Alamos National Laboratory. I feel uncomfortable to write a list of names mainly because so many people were important during the course of my endeavor that it seems dangerous to try to enumerate them all and perhaps leave someone out by accident. I am certain my friends know this acknowledgement is for them. Thank you so much and I hope I see you all again soon and have more time to spend with you now that I no longer have a dissertation to write.

Table of Contents

List of Tables

List of Figures

Abstract

Geoscientists have used machine learning for at least three decades and the applications spam many fields, from seismic processing and interpretation, to remote sensing classification, to analysis of well log data, among many others. More popular in some fields (e.g. seismic interpretation, remote sensing analysis) than others (e.g. paleontology), machine learning tools can leverage research in different areas of geoscience. Although machine learning is becoming more popular in different fields of geoscience, some concepts of more modern applications, convolutional neural networks in particular, are still vaguely understood by non-practitioners. I present some of the key concepts of machine learning with more details on the foundations of convolutional neural networks and some techniques that can help better understand convolutional neural networks behavior. I then present five case studies, mostly using convolutional neural networks and transfer learning. Transfer learning is a methodology that allow us to repurpose filters created by convolutional neural networks on a primary task to perform a secondary task. The five case studies start with a broader application of convolutional neural networks for different geoscience images, including thin-sections and core photographs. Then I present a how to perform core classification using convolutional neural networks. Next, how microfossils can be classified by the same methodology. I present a more detailed analysis of transfer learning using different remote sensing datasets. In the final case study, I show applications of supervised learning techniques to help forecast Megaelectron-Volt electrons inside Earth's outer radiation belt. I conclude the dissertation with a summary and comments on the expectation of future research.

Introduction

This dissertation compiles the work I developed during my years as a PhD student at the University of Oklahoma and also includes some of the research I conducted during an internship at the Los Alamos National Laboratory in Los Alamos, NM. This dissertation is organized in different chapters that are independent among themselves, but connected by the machine learning methodologies, used to address different research questions. The machine learning method used for the great majority of applications I present in this dissertation is convolutional neural network. All chapter topics are of interest to some field of geoscience, however the scale changes greatly: from the identification of microfossils using thin-section images to the forecast of space weather storms, including topics in core analysis and seismic facies classification. During the past few years, I had the grateful opportunity to collaborate with several amazing co-authors. Thus, as the chapters are organized based on journal papers published/submitted, most of the dissertation maintains first person plural. Despite great help from all co-authors, I was responsible for the totality of study design, implementation development, as well as the great majority of writing and figure creation.

Chapter 1 serves as an introduction to the topic of machine learning, especially convolutional neural networks, for the field of geosciences. This chapter presents the main terminology used in machine learning as well as the fundamentals of neural networks and convolutional neural networks.

Chapter 2 is presented as it was published in The Sedimentary Record (Pires de Lima et al., 2019a), which itself was based on an EAGE expanded abstract (Pires de Lima et al., 2019e). This chapter shows the results of the application of transfer learning to different sets of geoscience images, without delving into the details and challenges encountered in each one of the tasks. The application presented in chapter 2 shows the potential of the use of convolutional neural networks for different fields of geosciences. This chapter is also presented in my Data Science and Analytics Master's Thesis (Pires de Lima, 2019).

Chapter 3 is presented as it was published in Interpretation (Pires de Lima et al., 2019f). A preliminary piece was published in AAPG's Explorer (Pires de Lima et al., 2018). This chapter shows the results of the application of transfer learning to classify lithofacies from a core from the Mississippian limestone and chert reservoirs in the Anadarko Shelf, Grant County, Oklahoma.

Chapter 4 chapter was reworked and is presented here as and it was when resubmitted for publication in a sedimentary geology journal. This study was originated by a questioning from Dr. Lynn Soreghan during Murphy Cassel's Master's Thesis defense about fossil attributes. High quality thin-sections of Fusulinid specimens are hard to obtain and can provide invaluable information that can be used for dating through biostratigraphy as well as paleoenvironmental conditions, yet access to experts to experts can lead to an overlook of such microfossils. The results presented in this chapter show that convolutional neural networks can be used to classify Fusulinid thin-sections.

Chapter 5 presents more details of the use of transfer learning making use of remote sensing data. I use different network structures, optimizers, and datasets to evaluate the performance of transfer learning versus training randomly initialized weights. This chapter will be submitted to a remote sensing journal.

Chapter 6 shows some of the research I develop during my internship at the Los Alamos National Laboratory. The work in chapter 6 aims to forecast relativistic electrons inside Earth's outer radiation belt. Relativistic electrons are high energy and have the potential to destroy satellite electronics. I use different supervised learning models to forecast the behavior of 1.0 Megaelectron-volt (MeV) trapped inside Earth’s outer Van Allen belt. Interestingly, the results show how linear models are well capable to model most of the 1.0 MeV fluctuation. The work presented in this chapter was submitted to a space weather journal and the preliminary results are published as abstract in Pires de Lima et al. (2019b). The pre-print version is already available (Pires de Lima et al., 2019c). Unrelated to space weather, the work I developed during my internship at Los Alamos National Laboratory using simulation data for CO2 capture studies was published as two expanded abstracts (Pires de Lima et al., 2019d; Pires de Lima and Lin, 2019).

References

Pires de Lima, R., 2019. Petrographic analysis with deep convolutional neural networks. Master's Thesis. University of Oklahoma.

Pires de Lima, R., Bonar, A., Coronado, D.D., Marfurt, K., Nicholson, C., 2019a. Deep convolutional neural networks as a geological image classification tool. Sediment. Rec. 17, 4–9. https://doi.org/10.210/sedred.2019.2

Pires de Lima, R., Chen, Y., Lin, Y., 2019b. PreMevE 2.0: Neural Network Based Predictive Model for MeV Electrons in Earth’s Outer Radiation Belt, in: AGU Fall Meeting 2019. American Geophysical Union (AGU).

Pires de Lima, R., Chen, Y., Lin, Y., 2019c. Forecasting Megaelectron-Volt Electrons inside Earth’s Outer Radiation Belt: PreMevE 2.0 Based on Supervised Machine Learning Algorithms. ArXiv: physics.space-ph/1911.01315

Pires de Lima, R., Lin, Y., 2019. Geophysical data integration and machine learning for multi-target leakage estimation in geologic carbon sequestration, in: SEG Technical Program Expanded Abstracts 2019. pp. 2333–2337. https://doi.org/10.1190/segam2019-3215405.1

Pires de Lima, R., Lin, Y., Marfurt, K.J., 2019d. Transforming seismic data into pseudo-RGB images to predict CO2 leakage using pre-learned convolutional neural networks weights, in: SEG Technical Program Expanded Abstracts 2019. Society of Exploration Geophysicists, pp. 2368–2372. https://doi.org/10.1190/segam2019-3215401.1

Pires de Lima, R., Marfurt, K., Duarte, D., Bonar, A., 2019e. Progress and Challenges in Deep Learning Analysis of Geoscience Images, in: 81st EAGE Conference and Exhibition 2019. EAGE. https://doi.org/10.3997/2214-4609.201901607

Pires de Lima, R., Marfurt, K., Suriamin, F., Pranter, M., Soreghan, G., 2018. Convolutional Neural Networks - If they can identify an oncoming car, can they identify lithofacies in core? AAPG Explorer.

Pires de Lima, R., Suriamin, F., Marfurt, K.J., Pranter, M.J., 2019f. Convolutional neural networks as aid in core lithofacies classification. Interpretation 7, SF27–SF40. https://doi.org/10.1190/INT-2018-0245.1

Chapter 1: Elements of machine learning and convolutional neural networks for geoscientists and seismic interpreters

Rafael Pires de Lima[1,2], Kurt Marfurt[1]

[1]School of Geology and Geophysics, The University of Oklahoma, 100 East Boyd Street, RM 710, Norman, Oklahoma, 73019, USA

[2]The Geological Survey of Brazil – CPRM, 55 Rua Costa, São Paulo, São Paulo, Brazil

Preface

Although geoscientists have been using machine learning for decades, non-practitioners might feel discouraged use machine learning to facilitate their tasks due to a lack of familiarity with the process. In this first chapter I present some fundamental naming convention of machine learning elements as well as how deep neural networks, with details on convolutional neural networks, transform the data. This chapter will be submitted to a geoscience journal in the near future.

Abstract

Although machine learning is being marketed as a new solution to a wide range of problems, geoscientists have used use of machine learning for over three decades. Machine learning has been used to map seismic facies seen in 3D data, to predict missing well logs from those available, and to correlate well log data with seismic data. Driven by the rapid progress in computer vision coupled with the availability of affordable computer graphics processing units, deep neural networks techniques such as convolutional neural network (CNN) algorithms promise to help accelerate seismic interpretation, with applications ranging from semiautomated fault prediction to volumetric mapping of salt diapirs. In part due to its computational complexity, and in part due to a desire to aggressively market a new technology, CNN has generated perceptions ranging from "black box" to "magic". Our goal in this paper is lift the veil shrouding CNN and provide a tutorial that explicitly defines the assumptions and implementational details of CNN for the geoscience audience. Indeed, convolution forms the basis of most seismic processing algorithms while the closely related correlation process forms the basis of many well log analysis processes. We describe how the input geological and geophysical measurements are transformed by CNN to predict a known output for problems ranging from seismic facies analysis to core analysis. Rather than provide detailed case studies, we provide a broader description of the assumptions made, nonlinear processes used, promises for future applications, and potential pitfalls in CNN analysis.

Introduction

Machine learning (ML) techniques have been used by geoscientists for a variety of tasks. Lim et al. (1989) provided an early application applying both supervised and unsupervised machine learning techniques to synthetic aperture radar (SAR) to identify different Earth terrain components. Baldwin et al. (1990) used neural networks to classify lithofacies from well logs. Brown and Poulton (1996) used neural networks to detect objects and estimate their depth using electromagnetic and magnetic data. Cracknell and Reading, (2014) presented a comparison of ML techniques for geological mapping using remote sensing data. The list of applications of ML using seismological data is vast (e.g DeVries et al., 2018; Perol et al., 2018; Sinha et al., 2018; Ren et al., 2019), as is the list of applications using seismic data. Many authors used ML applied to seismic attributes to highlight seismic facies (e.g. de Matos et al., 2007, 2011; Roden et al., 2015; Qi et al., 2016; Zhao et al., 2016, 2018; Lubo-Robles and Marfurt, 2019) with a recent increase in the use of DNNs and CNNs to highlight seismic facies, to invert for impedance, to denoise data and others (e.g. Di et al., 2018; Li et al., 2018; Waldeland et al., 2018; Alfarraj and AlRegib, 2019; Di et al., 2019a; Mustafa et al., 2019; Pham et al., 2019; Zhao et al., 2019). Much like in other fields of science, geoscientists now are witnesses to an increase use of deep neural networks (DNN) and convolutional neural networks (CNN) applications.

LeCun et al. (2015) provided details of DNNs and CNNs. They reported that the work presented by Krizhevsky et al. (2012) was a breakthrough that caused the rapid embracing of DNNs and CNNs by the computer vision community. Krizhevsky et al. (2012) used CNN in a structure commonly referenced to as AlexNet on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC, Russakovsky et al., 2015), achieving new state-of-the-art accuracy levels.

The advent of DNN, CNN, and backpropagation allowed researchers to use “raw” data instead of features, commonly named attributes when referring to seismic data, as input to ML models. In their earlier work, Zhao et al., (2015) used seismic attributes to help recognize seismic facies in 3D seismic data volume acquired over the Canterbury Basin, offshore New Zealand. More recently, Zhao (2018) applied CNN directly to the seismic amplitude volume to define seismic facies in the F3 seismic survey acquired in the North Sea, offshore Netherlands. Wu (2017) developed a directional structure-tensor-based coherence algorithm to highlight faults and channels in seismic data; whereas Wu et al. (2019) used a 3D CNN model to identify faults. Other examples of the use of DNN or CNN for seismic data include Araya-Polo et al. (2017) who used DNN to learn a mapping relationship between the synthetic seismic shots input and the spatial points indicating fault presence. Waldeland et al. (2018) used a 3D CNN model to map salt bodies. Wang et al. (2019) used CNN for seismic data interpolation. Yang and Ma (2019) used CNN to invert pre-stack seismic data for velocity.

DNN and CNN has increased in geoscience applications beyond seismic interpretation. Wang et al. (2018) used a CNN to reconstruct high-resolution porous structures based on low-resolution micro X-ray computed tomography images and high-resolution scanning electron microscope (SEM) images. Pires de Lima et al.(2019a) used CNN to classify images of core data. Valentín et al. (2019) used ultrasonic and micro resistivity borehole image logs as input to a CNN models to predict the probability that any given sample belongs to one of their lithofacies classes. Pires de Lima et al. (2019b) and Ran et al. (2019) used CNN to classify rock images, among other geoscientific images. Duarte-Coronado et al. (2019) used CNN to classify the level of porosity of thin-sections images from the Mississippian strata in the Anadarko basin, Oklahoma.

The AAPG and SEG journal Interpretation and the SEG journal The Leading Edge recently held special ML issues (Di et al., 2019b; Jayaram et al., 2019; Shaw et al., 2019), while a special issue of Geophysics also addresses ML applications (Hu et al., 2019). This level of activity reflects the increased promise of ML techniques in addressing geoscience problems. In spite of these recently publications, we have found that many key aspects of ML terminology are only vaguely understood by those of us not in direct contact with ML procedures. Rather than showing a new application of some ML methodology, in this tutorial paper our goal is to define concepts that will allow the geoscience community to better understand the naming conventions as well as some of the assumptions, methods, and limitations of recent ML applications, with a focus on CNNs.

We begin our paper by defining the most common terms used by the machine learning community. We then define the components of a neuron and how multiple neurons can be linked to form a DNN layer, as well as how convolutions can be combined to form a CNN layer which in turn are linked to form a network. Next, we describe what happens to the input data during training in a simple CNN model with only a few layers, focusing on some key techniques that provide insight into how CNNs generate their results. With this background, we apply the CNN workflow to a seismic classification problem from the Gulf of Mexico. We conclude with a summary of the strengths and limitations of the method.

Terms commonly used in machine learning literature

This section aims not to be exhaustive, but rather to provide the reader with the necessary key terms often used in ML studies. We refer the reader to Google's “Machine Learning Glossary" (https://developers.google.com/machine-learning/glossary, accessed November 2019) as well as Pedregosa et al.'s (2011) scikit-learn website for a much more extensive glossary of terms as well as information about many other ML methodologies.

Chapelle et al. (2006) break ML into two large families: supervised learning and unsupervised learning. Ayodele (2010) expands these two categories by adding reinforcement learning and semi-supervised learning among others. As most applications of ML in geoscience rely on either unsupervised or supervised learning, these are the techniques we address here.

Chapelle et al. (2006) define a set $\mathbf{X} = (x_1, x_2, \dots, x_n)$ of n samples, or examples, where $x_i \in \chi$ for all $i \in \{1, 2, \dots, n\}$. It is generally assumed that the samples are drawn independently and identically distributed (i.i.d.) from a common distribution in χ . It is often convenient to define the $(n \times d)$ matrix $\mathbf{X} = ({x_i}^T)^T{}_{i \in n}$ that contains the samples as its rows where d is the dimension of the vectors x. The goal of unsupervised learning is to find meaningful structures in the data **X**. In other words, in unsupervised learning, we provide the ML algorithm with input data, also referred to as features, but with no defined target **Y**. Practices of dimensionality reduction such as principal component analysis (PCA, e.g. Guo et al., 2009; Hu et al., 2017; Pires de Lima and Marfurt, 2018) and independent component analysis (ICA, e.g. Honório et al., 2014; Lubo-Robles and Marfurt, 2019) are unsupervised learning practices. Kohonen's (1990) self-organizing maps (SOM, e.g. Coléou et al., 2003; de Matos et al., 2007; Cracknell et al., 2015; Zhao et al., 2016, La Marca-Molina et al., 2019) and Bishop et al.'s (1998) generative topographic mapping (GTM, e.g. Roy et al., 2014; Qi et al., 2016) can also be interpreted as a dimensionality reduction technique. In practice users usually interpret the map projected in the lower dimensional to generate clusters.

Chapelle et al. (2006) further state that, the goal of supervised learning is to learn a mapping from x to y, given a training set made of pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$. Note both input features x as output y can be multidimensional. Like in unsupervised learning, a standard requirement is that the pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ are sampled i.i.d. from some distribution spamming over $X \times Y$ space. y are called the labels or targets of the samples. When the labels are continuous, the ML task is called regression. When the labels belong to a finite discrete set, the ML task is called classification. An example of regression is the inversion of seismic amplitude data to estimate velocity (e.g. Yang and Ma, 2019). An example of classification is the estimation of a seismic facies from seismic amplitude data (e.g Waldeland et al., 2018; Wu et al., 2019)

Algorithms that use only samples $\boldsymbol{X}$ are defined as unsupervised learning. A common output of unsupervised methods is a colored volume where different colors are assigned to different clusters. Then after the clusters have been generated, the interpreter uses well data or principals of seismic geomorphology to assign a particular color to a specific geologic class. In contrast, algorithms that use a set of inputs paired with their labels, $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, fall under the realm of supervised learning. Chapelle et al. (2006) describes semi-supervised learning algorithms which require more details to be defined.

The main characteristic of ML algorithms is their ability to improve their performance, e.g., to classify each sample based on the features provided to the ML algorithm, through automatic analysis of data. Such automated analysis is often called “training” with the data used during this analysis called “training data”. Training finds the most appropriate weights or internal parameters for the ML model. Note that both unsupervised and supervised ML models need to be trained, and generally will attempt to either minimize a loss (the sum of errors of the training data) or maximize an expectation function, called the “objective function”. The internal variables of a ML model are named “parameters”, or “weights”, and are automatically updated during the training process. In contrast “hyperparameters” are defined by the user. For example, in K-means clustering (Hartigan and Wong, 1979) the number of clusters *K* is a hyperparameter, as well as the number of iterations the model is allowed to update itself. The cluster centers or means of the samples of each cluster are parameters. The user can choose the number of clusters, but the clusters centers are updated automatically by the model. In SOM and GTM the number of neurons and the dimensionality of the latent space are hyperparameters.

One common problem of ML models is their tendency to overfit the training data. Overfitting essentially means the model matches the training data so closely that it fails to perform correct predictions of new data not used in the training step. Thus, ML users habitually separate the data into three data sets: training, validation, and test data sets. Validation sets are sometimes called “development” or “dev sets”, validation and test sets are sometimes called “holdout” sets. Holdout sets are useful to determine if the ML model is overfitting the training data and if so, the level of overfitting. Validation sets are used during hyperparameter tuning, i.e. the process of choosing the hyperparameters that generate the best results. Ideally the test set is used only when the entire training and hyperparameter tuning process has been completed. Some ML users employ k-fold cross-validation in their analysis. In k-fold cross-validation, different subsets of the training data are selected to be used as the validation set (e.g. Hampson et al., 2001; Russell, 2004). For example, 5-fold cross-validation fits a ML model five different times using different subsets of training and validation data. The test set provides the most important metrics for ML models as they represent data unseen by the model.

The performance of the ML model is evaluated with different metrics depending on the ML model's task. Unsupervised learning clustering tasks use both internal validation metrics and external validation metrics (Palacio-Niño and Berzal, 2019). Internal validation metrics focus on measuring the cluster's cohesion (how similar the samples inside the cluster are to each other) and separation (how different clusters are from each other). External validation metrics use extra information to evaluate the quality of the clustering, for example how well the clusters align with known classes. Although dimensionality reduction techniques can be evaluated based on their ability to reconstruct the data, Palacio-Niño and Berzal (2019) find that determining the quality of the results of other clustering algorithms to be a very difficult problem. Kleinberg (2003) defined essential properties a clustering algorithm should satisfy; he then proved that no clustering algorithm can simultaneously satisfy all of them. In spite of these theoretical challenges, geoscientists still make use of imperfect clusters as a tool in a larger interpretation workflow.

In contrast to unsupervised learning, supervised learning models are relatively easier to have their performance assessed. Metrics for regression tasks include the common mean squared error, and the R2 score. R2 is given by:

$$R2 = 1 - \frac{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(y_i - f_i)^{\boldsymbol{2}}}{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(y_i - \bar{y})^{\boldsymbol{2}}} \tag{1}$$

where y_i is the true target value, $\bar{y}$ is the mean of $\boldsymbol{y}$, and f_i is the value provided by the ML model. R2 does not have a lower bound, and have a maximum (perfect) score of 1.0, meaning all of the variance of the predicted value was explained by the input data and the model, or that $\boldsymbol{f} = \boldsymbol{y}$.

Metrics for classification tasks generally include counting the number of total and incorrect predictions performed by a ML model. Many users define true positive (TP), true negative (TN), false positive (FP), and false negative (FN) to compute metrics. These are binary, positive and negative, but can also be generalized for multi-class problems. TP corresponds to a sample in which the model correctly predicted the positive class. TN corresponds to a sample in which the model correctly predicted the negative class. FP corresponds to a negative sample that the model incorrectly predicted as positive. FN corresponds to a positive sample that the model incorrectly predicted as negative. This information can be used to compute other metrics. For example, the precision, given by

$$Precison = \frac{TP}{TP + FP}$$

shows the frequency in which the model correctly predicted the positive class. Recall

$$Recall = \frac{TP}{TP + FN}$$

shows the proportion of TP correctly predicted. The accuracy and the confusion matrix are perhaps the easiest metrics to understand. The accuracy is given by the ratio of correct predictions over the total number of predictions. The confusion matrix is a table that summarizes the classification results comparing each one of the classes simply by using one axis of a matrix as the label that the model predicted, and the other axis as the actual label (the “true” value”).

Note all these metrics, both for unsupervised and supervised learning, can be applied to any of the sets (training, validation, test); however, the most important results are usually the ones from the test set.

Convolutional layers

The fundamental element of DNNs is the neuron. Neurons are systems that accept one or more input values and outputs a single value based on a function applied on the weighted sum of the inputs. The function applied on the weighted sum is generally referred to as activation function and is typically nonlinear in ML applications. A bias term is always added to the weighted sum, but we omit them for simplicity in this manuscript. Neurons can be organized into layers. DNNs are usually defined as multilayer neural networks with two or more hidden layers. Hidden layers are the layers between the input and output layers. Kim and Nakata (2018) and Russell (2019) discussed the similarities and differences between DNNs and geophysical inversion problems such as the computation of deconvolution operators.

Despite several variations on CNN architectures, all CNNs rely on the fundamental convolutional kernel. Convolution operates on two functions, one generally interpreted as the “input”, and the other as a “filter”. The filter is also referred to as “kernel”. The values of the kernels are the weights or parameters of a CNN. The kernel is applied to the input, producing an output. Just like neurons, a set of convolutional kernels can be combined into layers. Figure 1 provides a visual representation of DNNs and CNNs. Convolutions in ML are actually referred to as cross-correlations in signal processing, and such subtle difference is in fact irrelevant to the end results. In signal processing, the difference between convolution and cross-correlation is simply due to the fact that a convolution is actually cross-correlation with the reversed kernel, and cross-correlation works essentially as a sliding dot product of the input with the kernel. During the training stage, the values of kernels are updated in such a way that the output generated by the CNN is more similar to the desired label, thereby minimizing the objective function.

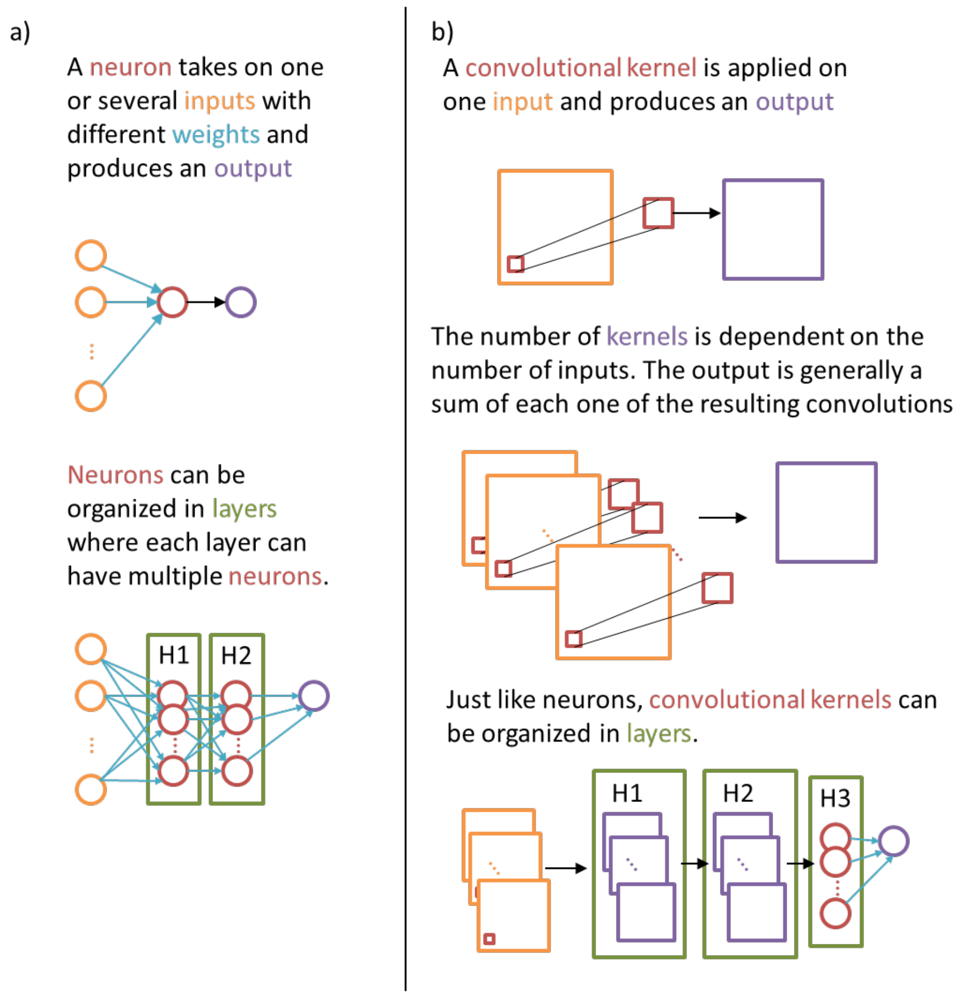

a)
A neuron takes on one or several inputs with different weights and produces an output
Neurons can be organized in layers where each layer can have multiple neurons.
H1
H2
b)
A convolutional kernel is applied on one input and produces an output
The number of kernels is dependent on the number of inputs. The output is generally a sum of each one of the resulting convolutions
Just like neurons, convolutional kernels can be organized in layers.
H1
H2
H3

Figure 1: Visual representation of a DNN and a CNN showing the main elements of each structure. (a) Cartoon showing a single neuron that applies an activation function to the weighted sum of the inputs and produces an output. We can then organize layers with multiple neurons, each neuron generally applying different weights to the set of inputs and producing different outputs. The output of a neuron can then be passed to the following layer, which can be either the next hidden layer or the final output. H1 and H2 represent the first and second hidden layers in this toy example. (b) Visual representation of a 2D CNN. Starting with the linear component, a convolution is applied to an input and produces an output. Next, an activation is applied to the output. Note that the number of convolutional kernels is equal to the number of input channels. Just like the neurons in (a), convolutional kernels can be organized in layers. Again, H1 and H2 represent respectively the first and second hidden layers in this toy example. For classification and regression tasks, the final layers of CNNs are usually the same as the ones used in DNNs, with layers containing neurons. The final layers of CNNs used for segmentation tasks are usually

convolutional layers as well. Note we choose to represent 2D CNNs as they are more common, however the idea is the same for 1D or 3D CNNs as well. Seismic facies classification is usually performed using 3D CNNs.

Note, as Figure 1 shows, that the CNN kernel applies different convolutional kernels to each one of the input channels. We use Figure 2 to illustrate the concept of a channel in ML using a color image of a *Triticites tomlinsoni* from the Waddell (1966) collection stored at the Sam Noble Museum - Oklahoma's Museum of Natural History. Colors in 2D images are often represented as a combination of red, green, and blue channels. Figure 2 shows the intensity for each one of the color components. Although our example is a 2D image, the concept expands to 1D and 3D as well. For example, a 3D CNN for facies segmentation using seismic amplitude and the envelope seismic attribute input volumes would be a two channel input data. Figure 3 shows how different convolutional kernels affect the input image. Other examples of the same concept are provided in the supplemental material.

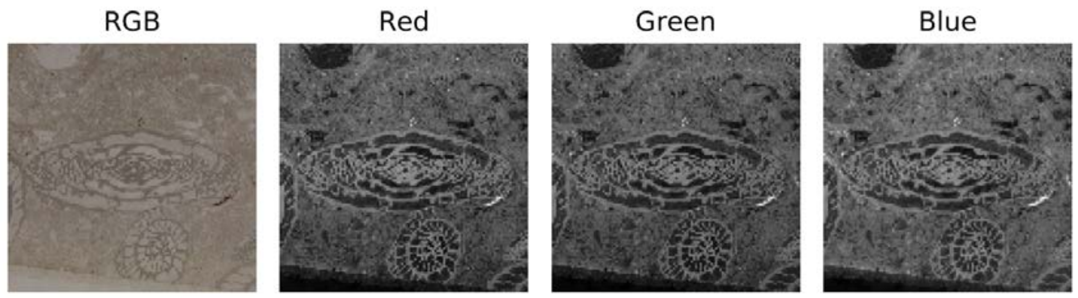


Figure 2: Decomposition of a RGB image into its red, green, and blue components. For red, green, and blue, low values are white and high values are black. Each one of the color components is referred to as a "channel". Each channel is composed of 299 x 299 pixels.

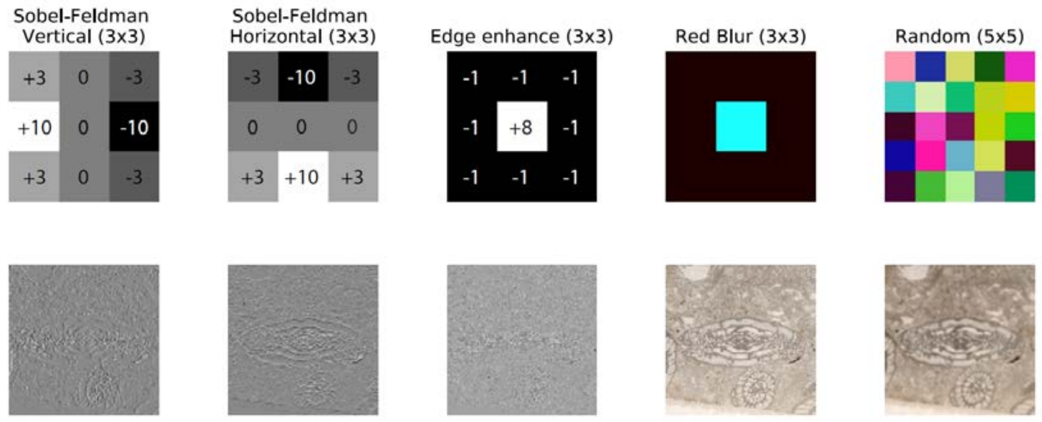


Figure 3: Representative output of different kernels applied to the three-channel input image shown in the previous figure. Each of the convolutional kernels (top row) needs to be three channels deep. The bottom row shows the result of the original image with each one of the convolutional kernels displayed on the top row. For the sake of visualization, we keep the outputs of each one of the channels separated; usually CNNs would sum the three channels into a single channel output. Note the images are not to scale – 3 x 3 pixels in the top row are much smaller than 299 x 299 pixels in top and bottom rows. Sobel-Feldman and edge enhance kernels have the same value for the three-different channels, thus we provide the numbers that compose the filters in the image. The red blur kernel weights the red channel by a fraction of its inputs (1/9) while the other channels remain unchanged – the center pixel is 1.0 whereas the outer pixels are zero, thus the cyan color in the center and the almost black color outline. The random kernel is generated with 5 x 5 x 3 random samples extracted from a random distribution, thus the colors are a combination of such random values applied to the three-channels. For visualization, all values are linearly scaled to range from 0-255.

Springenberg et al. (2014) observed that CNNs frequently use alternating convolution and max-pooling layers followed by a small number of fully connected layers. Moreover, CNNs are typically regularized during training using dropout layers. Max-pooling are simple down-sampling steps in which the maximum value for each sub-window (containing multiple values) of a feature is used to represent the entire sub-patch, effectively reducing the feature size, Figure 2 shows a visual representation of max and average pooling. Dropout layers randomly select a percentage of their inputs to be ignored during the training phase, helping to prevent overfitting. Not specific to CNN models, Srivastava et al. (2014) showed that dropout improves the performance of neural networks on many supervised learning tasks such as speech recognition, and computer vision.

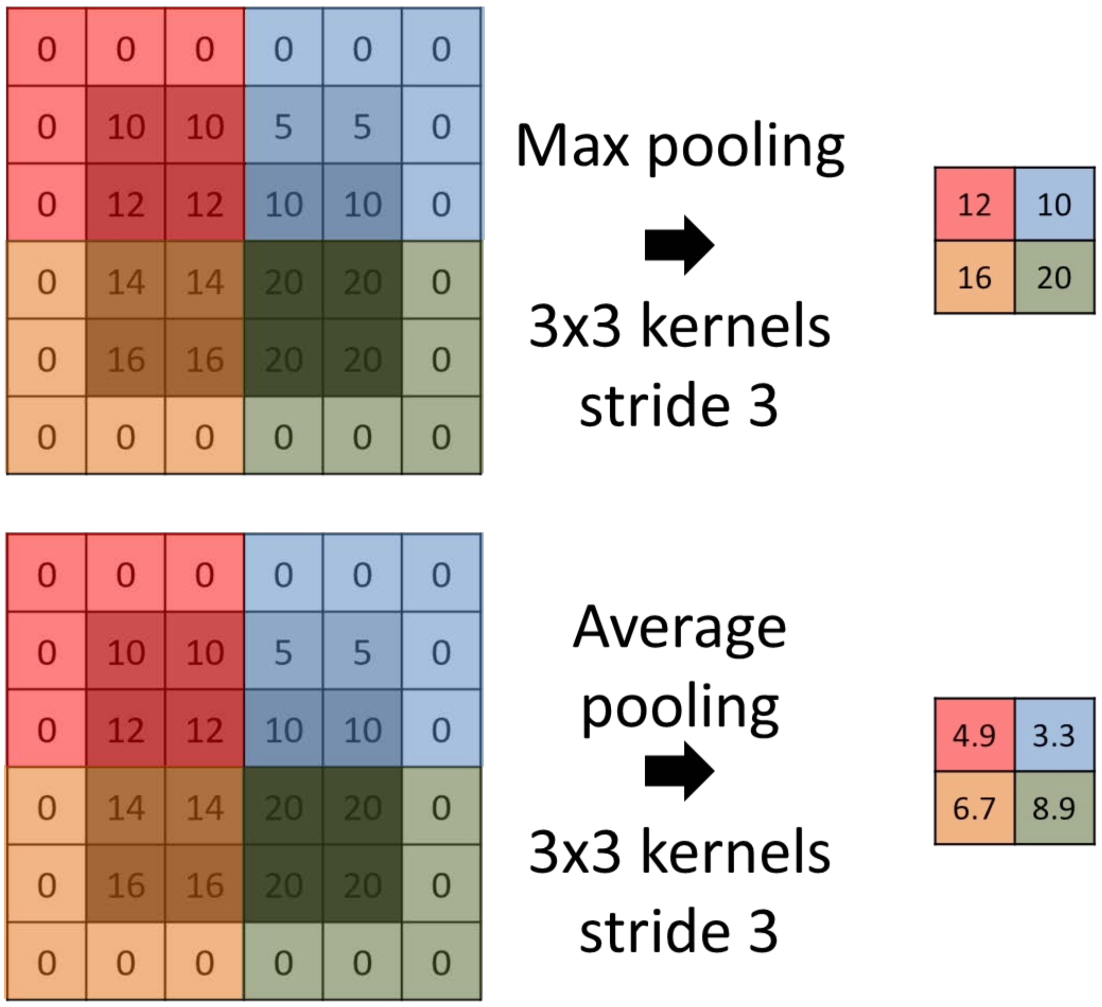


Figure 4: A representation of the max and average pooling operation, where the output is simply the maximum value of a subset of the input. Colors are used to facilitate visualization. Different strides and padding techniques can be used with pooling layers, as well as different statistics (minimum, average, median, or other statistical measures). We represent the pooling applied with a 2D input; however, the same technique can be applied to 1D or 3D inputs as well.

Backpropagation and training a simple CNN

LeCun et al. (2015) reported that the goal of researchers has been to replace hand-engineered features with trainable multilayer networks since the early days of pattern recognition

(Rosenblatt, 1957; Selfridge, 1958). However, in spite of its simplicity, the mathematical gradient descent solution technique was only recognized in the mid-1980s. LeCun et al. (2015) showed how backpropagation can be used to calculate the gradient of an objective function with respect to the inputs of multilayer neural architectures through the use of chain rule for derivatives.

We propose a simple experiment to further illustrate some of the key concepts of ML, to show how CNNs transform the data, and how backpropagation updates the weights in CNN. We create a simple dataset composed of 2D RGB images with three very different classes easily separable based on their image content. Each class is composed of only 16 samples. Samples for class "fossil" are images from Waddell (1966), samples for class "oil well" are images from PatternNet (Zhou et al., 2018), and samples for class "seismic" are created from inline, crossline, and time-slices of the Kora Survey (e.g Bischoff et al., 2017; Infante-Paez and Marfurt, 2017; Morley, 2018). To create the "RGB" seismic data, we simple select inline, crossline, and time slices. Figure 5 shows examples of the toy dataset we use. We rescale the images to 100 x 100 pixels, where we note the fossil class gets distorted as it was originally rectangular. The similarity between inline and crossline is so small that the images appear almost gray, time slices show slightly larger variation with some color highlighting some channels. Our goal is to train a CNN to be able to distinguish such toy data. Here we do not separate the data into training, validation, and test set, as we are mostly interested in observing how CNN changes the data rather than in properly training a model to classify images.

The CNN model we use is also simple. It is composed of an input layer receiving images with the shape 100 x 100 x 3, followed by a convolutional layer with eight filters of size 3 x 3, a max pooling layer with size 2 x 2 and stride two, another convolutional layer with 16 filters of size 3 x 3, a max pooling layer with size 2 x 2 and stride two, a convolutional layer with 32 filters of size 3 x 3, a max pooling layer with size 2 x 2 and stride two, and finally a fully connected neural layer with three outputs. The activation function of the convolutional layers is the rectified linear unit (ReLU), which is simply the half-wave rectifier

$$f(z) = max(z, 0).$$

The activation of the last layer with three outputs uses the softmax activation

$$f(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}.$$

Figure 6 shows a visual representation of the simple CNN we use. Dumoulin and Visin, (2016) presents details of the arithmetic useful for CNNs. Note we use “valid” padding, meaning we can only apply the filters where there is an input and we do not change the size of the filters, therefore the shape of the channels is reduced after each one of the convolutions. Max pooling with size 2 x 2 and stride two should reduce the shape of the channel exactly in half, but because the inputs are odd the last value is dropped. We use TensorFlow (Abadi et al., 2016) to implement the CNN model.

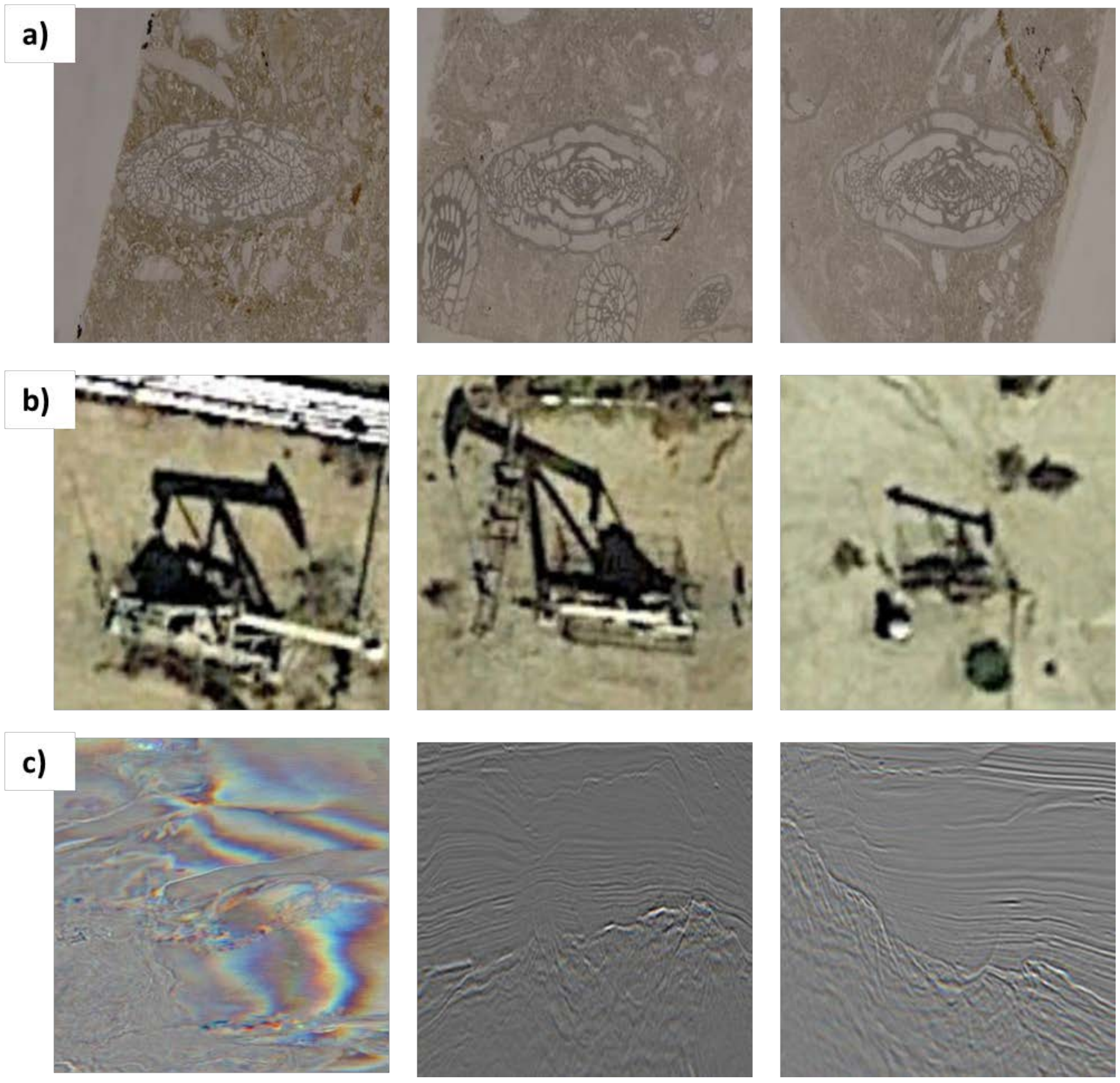


Figure 5: Examples of the toy dataset we use to illustrate CNN models. (a) shows examples of the class “fossil” coming from Waddell's (1966) collection, (b) shows examples of the class “oil well” coming from Zhou et al.'s (2018) PatternNet, and (c) shows examples of the class “seismic”, coming from the Kora 3D seismic survey. (c) shows color for the leftmost image as that is composed of a time slice whereas the other examples are vertical slices. The seismic data changes more rapidly in the vertical direction, therefore the changes are more visible when composing a three-channel image. Note the vertical slices are in fact colored and composed of three-channels, but the colors are almost imperceptible as the vertical slices present very similar content in their vicinities.

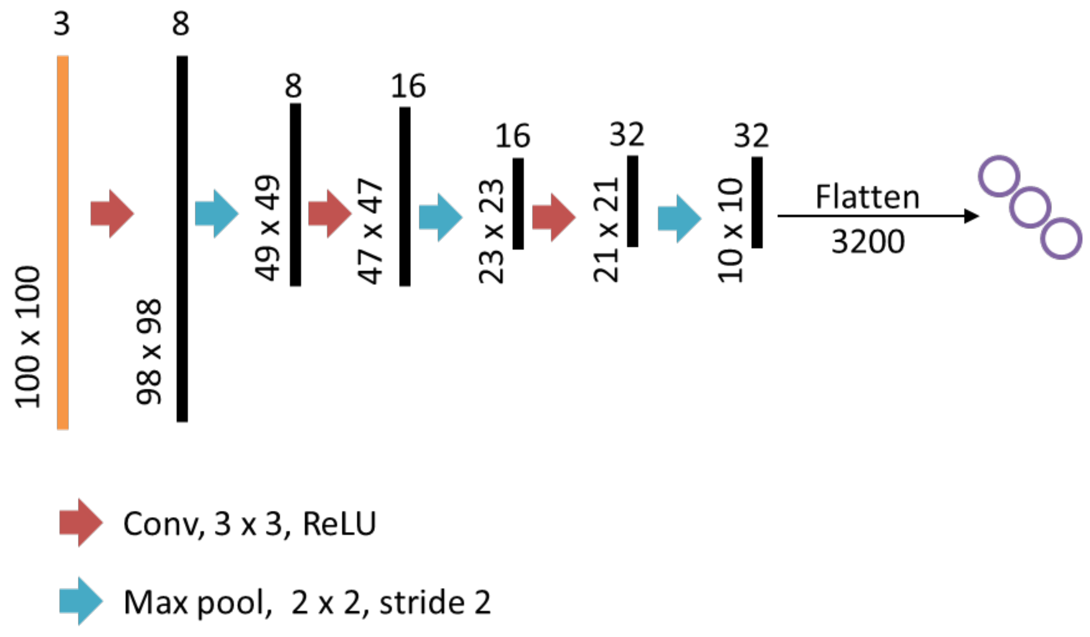


Figure 6: A representation of the simple CNN model we use. Numbers on top of the vertical bar indicate the number of channels whereas number on the side indicate the shape of each one of the channels for each particular layer. Pooling layers are operations that reduce the shape of the channel and do not need to be trained. Note in practice we flatten the 10 x 10 x 32 array into a 32,000 x 1 array to be used as input for the final layer.

With data and model ready, we need to specify the loss function and the optimizer to train the model. The loss function provides the model with a measure of how good or bad the model is performing. The optimizer is the algorithm to be used to reduce the loss, very generally a variation of stochastic gradient descent. We chose Kingma and Ba's (2014) Adam optimizer with the parameters defined by the authors. We use cross-entropy as the loss function. When we minimize the cross-entropy, we incentivize the CNN to increase the probability the sample has to be assigned to the class aligned with the label provided to that sample. We train the model for 25 epochs. An epoch is one pass in all data being used, 48 samples in this toy example case. The weights of the CNN model are randomly initialized with Glorot Uniform distribution (Glorot and Bengio, 2010), and our expectation is that training will modify the weights in such a manner that the CNN will predict the correct class for the samples we provide. Figure 7 shows one oil well sample and the output of each one of the convolutional layers, as well as the resulting classification provided by the CNN on the last layer. Figure 8 shows the results obtained after training the model for 25 epochs. The supplemental material contains extra examples for a fossil sample as well as one seismic sample. Video versions showing the evolution of the data transformation epoch-by-epoch is provided on online the online version of this manuscript.

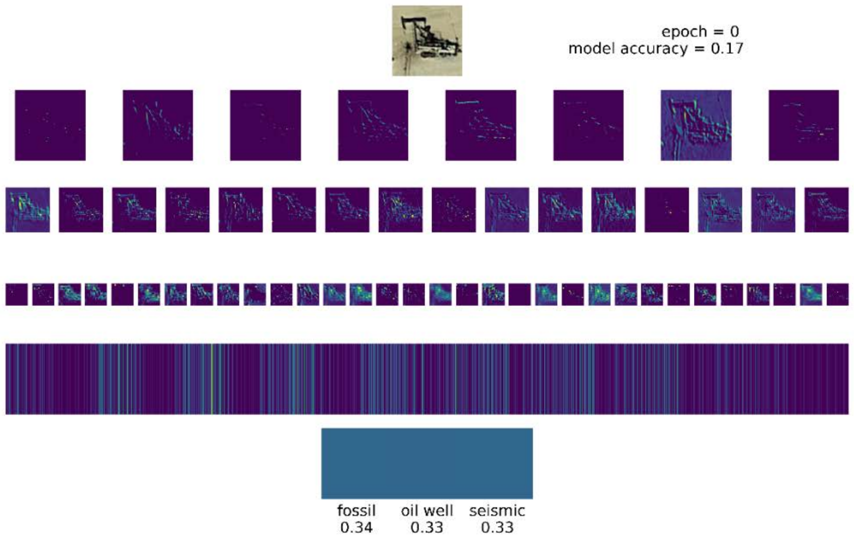


Figure 7: Example of how the data a transformed for different layers in our simple CNN model. When the model starts, the weights are randomly generated and the CNN assigns a fairly balanced probability the image can belong to each one of the classes. The image on top shows the input to the CNN. The first row shows the results of the first layer consisting of eight convolved outputs. The second row shows the results of the 16 filters of the second convolutional layer, after the first max pooling layer. The third row shows the results of the 32 filters of the third convolutional layer. The fourth row shows the flattened vector. Finally, the fifth and last row shows the results for each one of the three neurons on the final layer. Images not to scale. We refer the reader to the online version containing a video showing the evolution of these images for each epoch. Blue is used for small values and yellow for large. Images not to scale. This image shows the output of different filters, not the filters themselves.

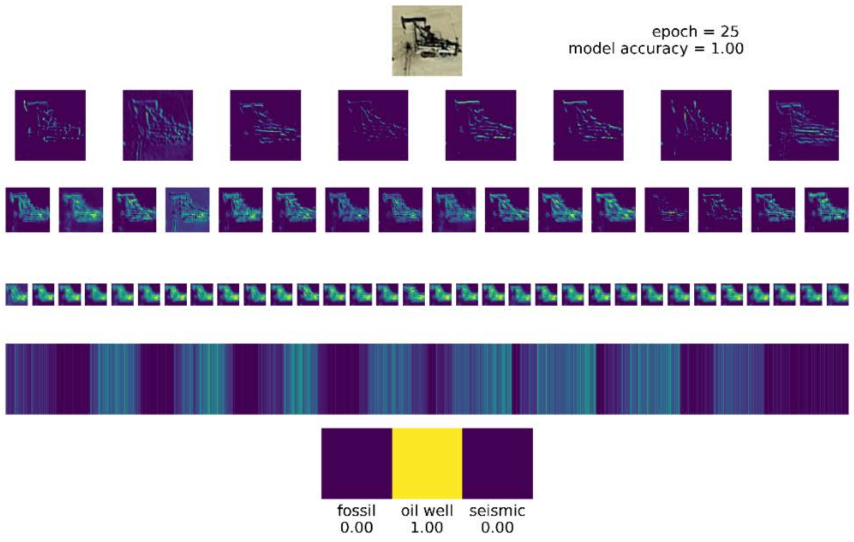


Figure 8: Same image as Figure 7 after the model is trained for 25 epochs. We refer the reader to the online version for a video showing the epoch-by-epoch evolution. The model assigns the sample to the correct class with a high probability. Blue is used for small values and yellow for large values. Images not to scale.

Despite Figure 8 showing how the data are transformed by CNN from the input to the final decision, it still hard to understand what leads the model to make the classification. There are many techniques that can be used to better understand CNN behavior. DNN interpretability is a research topic by itself (e.g. Simonyan et al., 2013; Zhou et al., 2016; Olah et al., 2017, 2018; Carter et al., 2019). We chose two techniques to help uncover what parts of the images are important for the final classification: saliency maps (Simonyan et al., 2013) and class activation mapping (CAM, Zhou et al., 2016). Both saliency maps and CAM are based on a similar principle based in the backpropagation concept. Saliency maps use backpropagation to calculate what regions of the input sample would create a larger difference in the final output, i.e. compute the gradient of the output (or an internal neuron) with respect to the input sample. Saliency maps can be improved with the implementation of guided-backpropagation (Springenberg et al., 2014). The idea of guided-backpropagation is to simply set the negative gradients to zero; saliency maps with guided-backpropagation usually produce more aesthetically pleasing results. CAM produces a heatmap showing what regions of the sample are most important for the final decision. CAMs, however, use the results of the last convolutional layer. The choice to use the last convolutional layer is to be able to utilize spatial information that gets lost in the final fully connected portion of a CNN. CAM averages the gradient of the filters on the final convolutional layer with respect to the input sample. We implement grad-CAM (Selvaraju et al., 2017) to show what areas are important for the classification performed by our CNN model on examples of our toy dataset. Grad-CAM is a CAM implemented with guided-backpropagation. Figure 9 shows the saliency map and grad-CAM computed for one sample of each one of the classes in our toy dataset.

To further illustrate the usefulness of saliency maps and grad-CAM, we now abandon our toy dataset and CNN model and use VGG16 (Simonyan and Zisserman, 2014) trained with the ILSVRC dataset. VGG16. VGG16 is a CNN model with 13 convolutional layers and three fully connected layers. Figure 10 show the saliency map and grad-CAM for two images never seen by the CNN model before. Note the model focused on the places useful to provide the classification for each one of the images, however due to the unusual cat pose, it fails to correctly identify the animal in Figure 10b. Visualization of saliency maps, grad-CAM and other variations can be very helpful to understand CNNs behavior. Zech et al. (2018) trained CNN models to identify patients with pneumonia based on their radiographies. The authors noted that the CNN learned to identify a metal token that radiology technicians place on the patient and actually focused on that information to provide the classification. CNN models then can actually use such information to predict disease such tokens strongly correlate with disease prevalence.

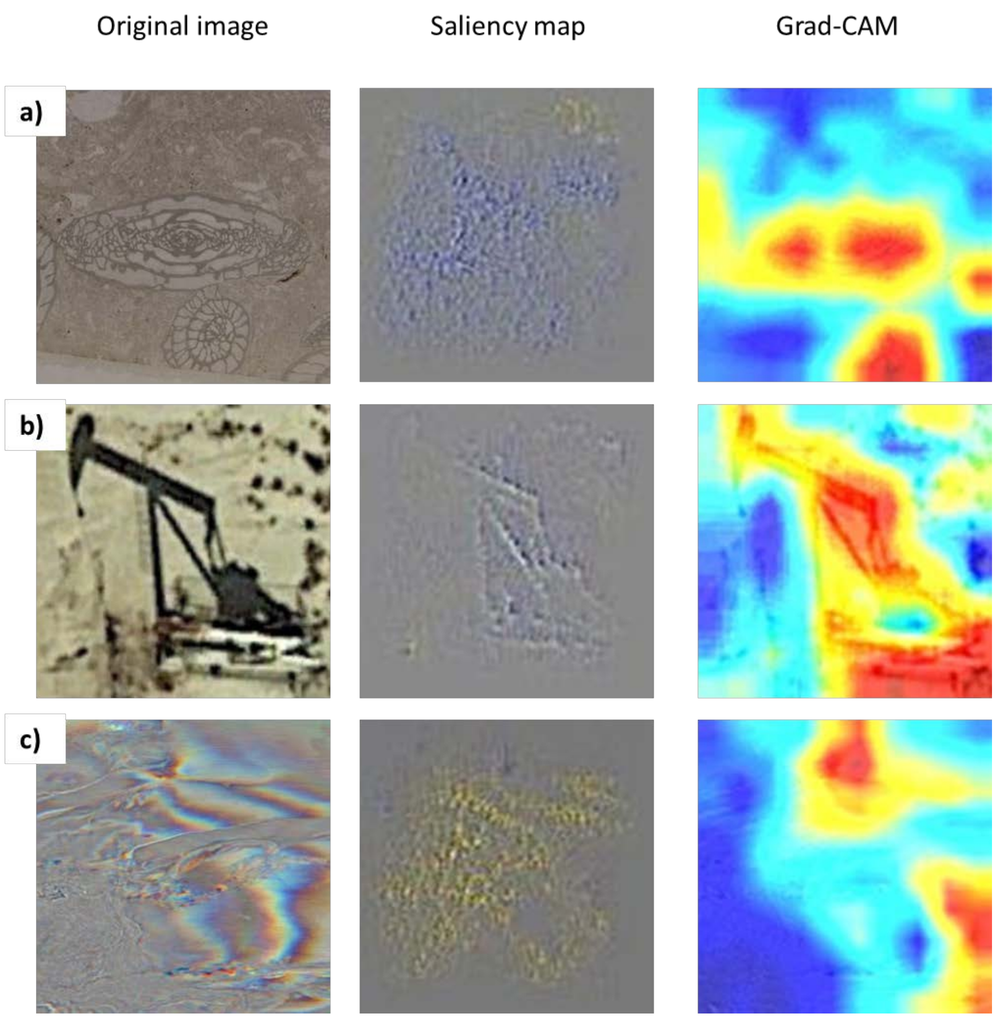


Figure 9: Samples from our small dataset, saliency map and grad-CAM generated with our simple model. (a), (b), and (c) show original image, saliency map computed with backpropagation, and grad-CAM. Saliency map shows gradient are spread for (a) and (c) and more focused on (b). Grad-CAM for (a) and (b) highlight areas that very clearly dominate the class in the picture. Note how the shadow of the oil well is important for its classification.

The dataset and the CNN we use to show the key components of CNN are small and simple. Compare our toy dataset composed of 48 samples with the MNIST dataset (LeCun, 1998) with tens of thousands of samples, or the previously referenced ILSVRC with hundreds of thousands of samples. Although greater development of DNN and CNN occurred concomitantly with the increase of available samples, research is also conducted with the objective to create robust CNN models using relatively few samples (e.g. Koch, 2015; Schroff et al., 2015; Rostami et al., 2019).

As our toy dataset is simple, a CNN with only three convolutional layers and one fully connected layer is enough to correctly classify all the samples. Compare this simple structure with VGG19 (Simonyan and Zisserman, 2014) that contains a CNN model with 16 convolutional layers and three fully connected layers, or Inception V3 (Szegedy et al., 2014, 2015) that uses blocks of convolutional layers with different filter sizes (5 x 5, 3 x 3, and 1 x 1), implements concatenation, and ends up being more than 40 layers deep. ResNets (He et al., 2016) and DenseNets (Huang et al., 2016) are hundreds of layers deep. Waldeland et al's. (2018) CNN model for seismic facies classification contains five convolutional layers. Wu et al's. (2019) FaultSeg, based on Ronneberger et al.'s (2015) U-net, used for seismic fault segmentation has tens of convolutional layers.

Activation maximization (e.g Olah et al., 2017) provides another way to try to understand what a CNN expects. Again making use of backpropagation, the activation maximization idea is to create an input to the CNN such that the output of a given neuron is maximized. In practice, we use an objective function to maximize the mean of a selected neuron’s output. The CNN then starts classifying a random noise image. Then, we compute the gradients using backpropagation to perform gradient ascent on the random noise image. Iteratively, we build an input that maximizes the neuron’s output. Figure 11 shows examples of activation maximization performed using VGG16 for the “ourzel” and “trilobite” classes. The online version of this manuscript contains videos showing how the images change from the starting noise to the final output. We implement a simple version that only tries to improve the visualization using total variation (TV) regularization. The choice of TV regularization is justified as activation maximization based on the optimization of an objective function that maximizes the output of a selected neuron as we implement can be highly affected by high frequency artifacts. In fact, generating appropriate activation maximization results is somewhat challenging. Olah et al. (2017) provided details on different approaches to generate better visualizations, such as regularizations, upscaling, and randomly shifting the image during the optimization process.

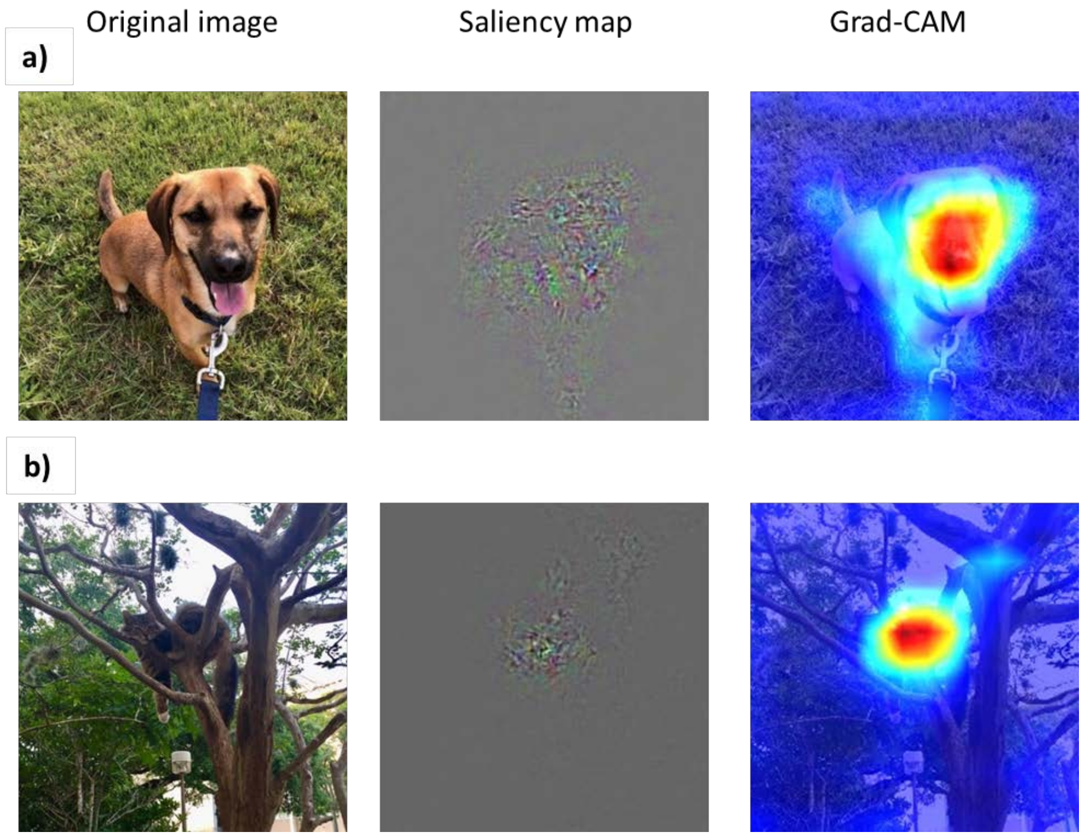


Figure 10: A picture of a dog, picture of a cat, saliency maps, and grad-CAM generated with VGG16. (a) and (b) show original image, saliency map, and Grad CAM. The original image (a) was classified as a Rhodesian ridgeback (0.21), whereas the second highest class for (a) was a redbone (0.16), two different dog breeds. Note ILSVRC has 1,000 classes and the probability to all classes sums to 1.0. The original image in (b) was classified as a spider monkey (0.69), the second highest class for (b) was a howler monkey (0.08). Saliency maps compute the gradient of the image with respect to the class assigned by the CNN model. We rescale the gradients from 0 to 255 to present them as color images. Grad-CAM is also based on gradients and computes the importance of the output filters towards the final decision. Grad-CAM implements guided backpropagation, in which negative gradients or gradients associated with a negative value of the filter are zeroed, rejecting elements that act negatively towards the decision, thus highlighting the most relevant zones for the model. Although the CNN model focuses on the areas in which the animals are located to provide the final classification, it is incapable of noticing the details of (b). This is very likely due to the fact that ILSVRC contains many pictures of monkeys on trees and few (if any?) pictures of cats on trees.

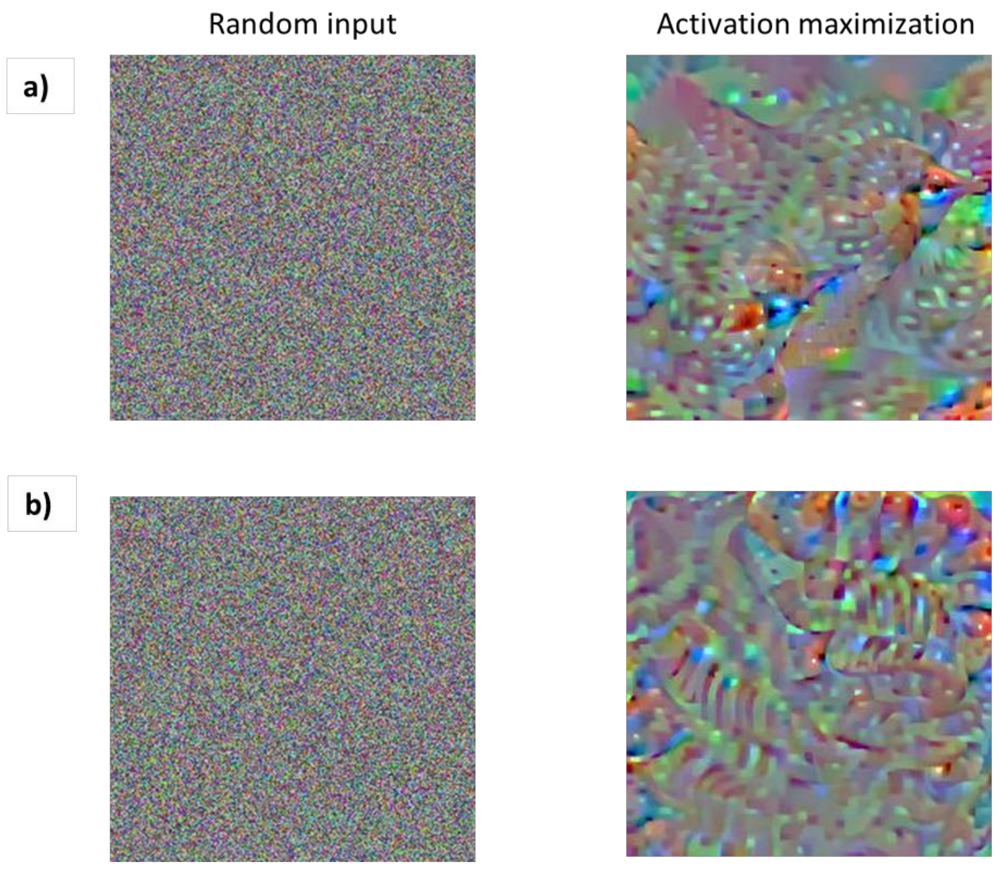


Figure 11: Results of activation maximization for ILSVRC for classes (a) “ourzel” and (b) “trilobite”. The input in this case are random values and the algorithm uses backpropagation to update the input in such a way that the output of a particular neuron is maximized. Note how the activation maximization of (a) seems to compose beak-like shapes whereas the activation maximization of (b) seems to be composed of “trilobite” textures.

CNN as a seismic facies classification tool

We now provide a simple application of CNN to classify seismic facies. We use the CNN model described by Waldeland et al. (2018) that uses as input 3D seismic volumes of 65 x 65 x 65 and outputs a single categorical value. Hence, we say this is a classification task. The objective of the CNN model is to use the input information, i.e. the mini 3D seismic cube, and to classify the label of the center voxel. Strictly speaking, the model will attempt to learn the mapping function of the input data to whatever we ask it to classify. Thus, the quality of the labels is very important now, as the labels are what the model needs to learn. For this task, we use a 3D seismic data acquired by PGS using towed streamer acquisition with two sources and three receiver cables with a maximum offset of 6000 m along the current offshore Louisiana shelf edge. The data map several salt diapirs and minibasins and we select a small subsection containing for our experiment. The volume we use for this experiment has a sample increment of 4 ms, with 37.5 m between lines and 12.5 m between CDPs, with respective dimensions of 1,996 x 326 x 682. The labels we provide for the CNN model are: conformal sedimentary layers (CSL), salt, and mass transport deposits (MTD). Figure 12 shows the total number of labeled samples for each one of the classes. To create this dataset, we simply selected examples of each one of the three classes from 21 different inlines. This "interpretation" process took roughly 20 minutes. For training, we select 20% of the data to be validation. We train the model for 15 epochs using Adam to minimize the cross-entropy loss. Our implementation was not optimized for data input-output management, but the training was completed in roughly one hour using a GeForce RTX 2060. Figure 13 shows how the accuracy and loss change for every training epoch. To obtain a seismic facies volume, we use the trained model to classify every fifth trace and we use bilinear interpolation on the time slices. Figure 14 shows examples of the labeled data we use to train the model as well as the classes provided by the CNN for the whole volume.

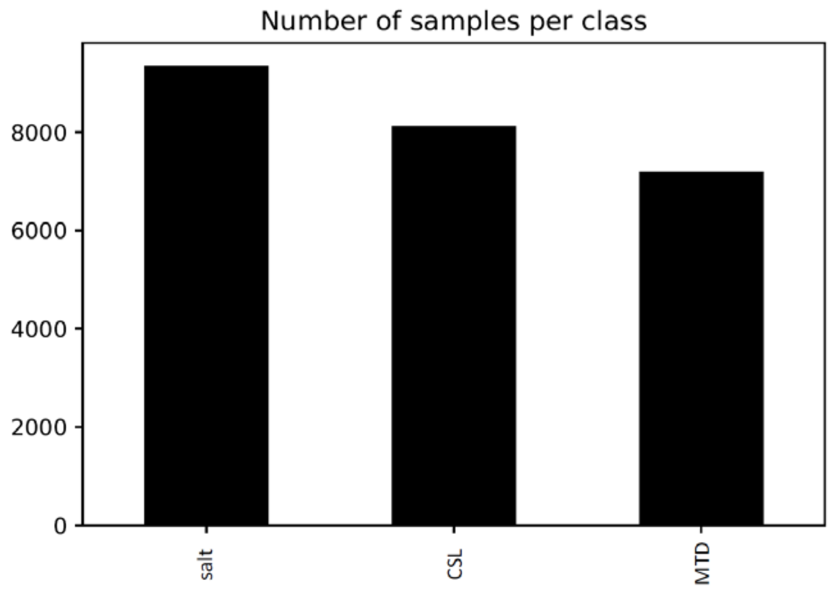


Figure 12: Histogram showing the number of samples for each one of the three target classes: salt, conformal sedimentary layers (CSL), and mass transport deposit (MTD). Note we tried to provide a somewhat balanced number of samples per class.

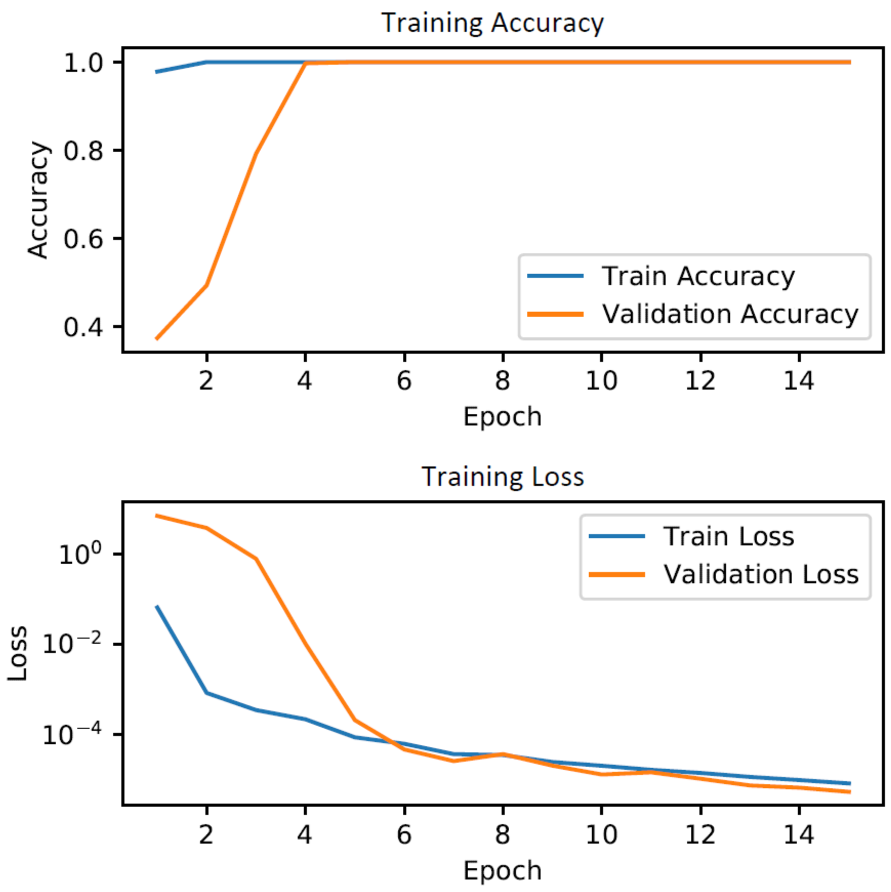


Figure 13: Model accuracy and loss evolution for 15 epochs of training. Note the accuracy for both training and validation sets reaches 1.0 at epoch four, whereas the loss continues to slowly decrease. The loss is plotted using logarithmic scale.

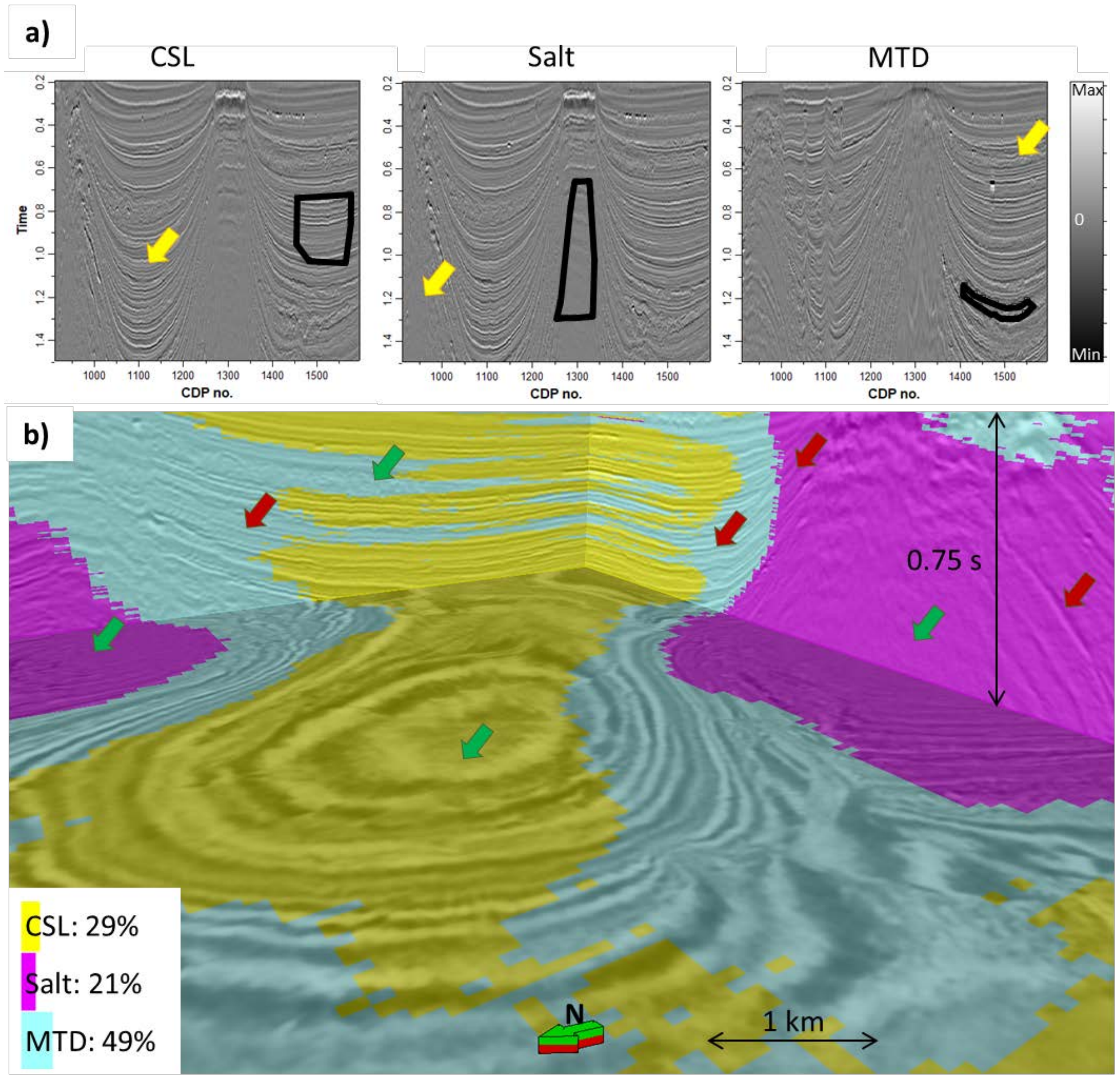


Figure 14: Human interpretation and CNN classification results. (a) shows seismic amplitude and examples of interpretation of conformal sedimentary layers (CSL), salt, and MTD in different inlines. The black polygons show the interpreted seismic facies that will be used to train the CNN; the yellow arrows in each one of the panels show examples of the same seismic facies present in that class not interpreted, therefore not using for training or validation. (b) The resulting interpretation of the trained CNN model away from the training data overlaid on seismic amplitude. Seismic interpreters will quickly notice areas wherein they agree with CNN provided facies, as well as many areas in which they would disagree. Green arrows indicate locations in which the model seems to correctly classify the seismic image as most interpreters would; red arrows indicate locations in which CNN provided facies predictions that needs to be improved. It is evident that the CNN is overpredicting the distributions MTD, very likely due to an insufficient number of training samples.

Limitations and suggestions for further study

Although we believe appropriate training data will allow CNN models to generate better results, and despite our decent results using very few training data for 3D CNN seismic facies classification task, at this moment we are unsure when should interpreters apply supervised learning in contrast to unsupervised learning algorithms. We hypothesize unsupervised learning techniques are more appropriate during exploration stages, when the interpreter is still unsure on what to expect of the data. Unsupervised learning methods can help provide a better overview of the data, highlighting different architectural elements “without being specifically asked to do so”. Then, for more detailed interpretation, supervised learning techniques including CNNs, random forest (e.g. Kim et al., 2019), probabilistic neural networks (e.g. Russell, 2004; Lubo-Robles et al., 2019), among others, are likely to provide more details for specific regions of interest.

Our implementation of saliency maps with simple backpropagation and Grad-CAM produced results too close to zero for the 3D CNN seismic classification example, which might be an indication of vanishing gradients. Currently we are unsure whether this unsatisfactory result, with gradients too close to numerical precision, is caused to our poor training data or due to some detail missing in our implementation. Nonetheless, such techniques can be useful to help interpreters understand why their models succeeds or fails in different contexts. Other estimations of uncertainty and interpretability can be even more helpful. For example, Wickstrøm et al. (2018) used measures of uncertainty based on dropout to map areas of low confidence during segmentation of polyp images. We believe the application of a similar methodology can be incorporated in many CNN models used for seismic facies classification, helping interpreters define areas of high and low confidence.

In this manuscript, our goal was to provide insight into CNN used in classification. Many applications of CNN with seismic datasets are actually based on CNNs using a 3D or 2D input to output a 3D or 2D output. In contrast, Waldeland et al.’s (2018) CNN used for 3D input and 3D convolutions to predict a single categorical output. Zhao (2018) compared results of seismic facies segmentation using a classification CNN that uses as input 2D data and outputs a single pixel answer, and a seismic segmentation that uses 2D data as output and outputs 2D data as well. Zhao (2018) found better results for the 2D input-2D output, with more continuous and less noisy results. Wu et al.'s (2019) and Shi et al.'s (2019) CNN for seismic facies segmentation, for example, are based on 3D input and 3D outputs

Conclusions

Despite the recent increase in interest in ML applications, the complexity and unfamiliarity with the technology has limited its acceptance by many practicing geoscientists. Geoscientists and engineers have years of training in deductive, physics-based reasoning. The classic approach is to apply a physics-based model to a process (e.g. wave propagation in the case of seismic observations), and find the parameters for the model (e.g. subsurface velocity) that predict the measured results. ML applications in general, and CNNs in particular, update their own internal parameters in the form of weights and non-linear components. These model parameters are generally based on statistics rather than on principals of physics and geology. Our objective in this tutorial has been to remove some of the black-box component from ML and show that the estimation of unknown parameters is closely related to the estimation of the deconvolution operators routinely used by seismic processors. We have shown step-by-step how a CNN transforms the data from input to output. We also show that with only a small amount of interpretation training data, that a CNN can provide useful results. These results will improve with better model architectures, better labels, and more experience.

Acknowledgments

Pires de Lima acknowledges CNPq (grant no. 203589/2014-9) for graduate sponsorship and CPRM for granting absence of leave. Funding for this project was provided by the industry sponsors of the OU Attribute-Assisted Seismic Processing and Interpretation Consortium. We would like to thank the PGS for providing a license to their data for use in research and education. We thank New Zealand Petroleum and Minerals for providing good 3D seismic data for the research community at large.

References

Abadi, M. et al., 2016, TensorFlow: A system for large-scale machine learning, in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16): 265–283.

Alfarraj, M., and G. AlRegib, 2019, Semisupervised sequence modeling for elastic impedance inversion: Interpretation, 7, SE237–SE249, doi:10.1190/INT-2018-0250.1.

Araya-Polo, M., T. Dahlke1, C. Frogner, C. Zhang, T. Poggio, and D. Hohl, 2017, Automated fault detection without seismic processing: The Leading Edge, 36, 208–214, doi:/10.1190/tle36030208.1.

Ayodele, T. O., 2010, Types of machine learning algorithms, in New advances in machine learning: IntechOpen.

Baldwin, J. L., R. M. Bateman, and C. L. Wheatley, 1990, Application of a neural network to the problem of mineral identification from well logs: Society of Professional Well Log Analysts.

Bischoff, A. P., A. Nicol, and M. Beggs, 2017, Stratigraphy of architectural elements in a buried volcanic system and implications for hydrocarbon exploration: Interpretation, 5, SK141–SK159, doi:10.1190/INT-2016-0201.1.

Bishop, C. M., M. Svensén, and C. K. I. Williams, 1998, GTM: The Generative Topographic Mapping: Neural Computation, 10, 215–234, doi:10.1162/089976698300017953.

Brown, M. P., and M. M. Poulton, 1996, Locating Buried Objects for Environmental Site Investigations Using Neural Networks: Journal of Environmental and Engineering Geophysics, 1, 179–188, doi:10.4133/JEEG1.3.179.

Carter, S., Z. Armstrong, L. Schubert, I. Johnson, and C. Olah, 2019, Activation Atlas: Distill, 4, e15, doi:10.23915/distill.00015.

Chapelle, O., B. Schölkopf, and A. Zien (eds.), 2006, Semi-Supervised Learning: Cambridge, MA, MIT Press.

Coléou, T., M. Poupon, and K. Azbel, 2003, Unsupervised seismic facies classification: A review and comparison of techniques and implementation: The Leading Edge, 22, 942–953, doi:10.1190/1.1623635.

Cracknell, M. J., and A. M. Reading, 2014, Geological mapping using remote sensing data: A comparison of five machine learning algorithms, their response to variations in the spatial distribution of training data and the use of explicit spatial information: Computers & Geosciences, 63, 22–33, doi:10.1016/J.CAGEO.2013.10.008.

Cracknell, M. J., A. M. Reading, and P. de Caritat, 2015, Multiple influences on regolith characteristics from continental-scale geophysical and mineralogical remote sensing data using Self-Organizing Maps: Remote Sensing of Environment, 165, 86–99, doi:10.1016/J.RSE.2015.04.029.

DeVries, P. M. R., F. Viégas, M. Wattenberg, and B. J. Meade, 2018, Deep learning of aftershock patterns following large earthquakes: Nature, 560, 632–634, doi:10.1038/s41586-018-0438-y.

Di, H., T. Zhao, V. Jayaram, X. Wu, L. Huang, G. AlRegib, J. Cao, M. Araya-Polo, S. Chopra, S. Al-Dossary, F. Li, E. Gloaguen, Y. Lin, A. Solberg, and H. Zeng, 2019, Introduction to special section: Machine learning in seismic data analysis: Interpretation, 7, SEi-SEii, doi:10.1190/INT-2019-0609-SPSEINTRO.1.

Di, H., D. Gao, and G. AlRegib, 2019, Developing a seismic texture analysis neural network for machine-aided seismic pattern recognition and classification: Geophysical Journal International, 218, 1262–1275, doi:10.1093/gji/ggz226.

Di, H., Z. Wang, and G. AlRegib, 2018, Why using CNN for seismic interpretation? An investigation, in SEG Technical Program Expanded Abstracts 2018: Society of Exploration Geophysicists, SEG Technical Program Expanded Abstracts, 2216–2220, doi:10.1190/segam2018-2997155.1.

Duarte-Coronado, D., J. Tellez-Rodriguez, R. Pires de Lima, K. Marfurt, and R. Slatt, 2019, Deep convolutional neural networks as an estimator of porosity in thin-section images for unconventional reservoirs, in SEG Technical Program Expanded Abstracts 2019: Society of Exploration Geophysicists, 3181–3184, doi:10.1190/segam2019-3216898.1.

Dumoulin, V., and F. Visin, 2016, A guide to convolution arithmetic for deep learning: ArXiv e-prints.

Glorot, X., and Y. Bengio, 2010, Understanding the difficulty of training deep feedforward neural networks, in In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics.

Google, 2019, Machine Learning Glossary: <https://developers.google.com/machine-learning/glossary/> (accessed November 3, 2019).

Guo, H., K. J. Marfurt, and J. Liu, 2009, Principal component spectral analysis: Geophysics, 74, P35–P43, doi:10.1190/1.3119264.

Hampson, D. P., J. S. Schuelke, and J. A. Quirein, 2001, Use of multiattribute transforms to predict log properties from seismic data: GEOPHYSICS, 66, 220–236, doi:10.1190/1.1444899.

Hartigan, J. A., and M. A. Wong, 1979, Algorithm AS 136: A K-Means Clustering Algorithm: Journal of the Royal Statistical Society. Series C (Applied Statistics), 28, 100–108, doi:10.2307/2346830.

He, K., X. Zhang, S. Ren, and J. Sun, 2016, Identity Mappings in Deep Residual Networks, in B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., Computer Vision -- ECCV 2016: Springer International Publishing, 630–645.

Honório, B. C. Z., A. C. Sanchetta, E. P. Leite, and A. C. Vidal, 2014, Independent component spectral analysis: Interpretation, 2, SA21–SA29, doi:10.1190/INT-2013-0074.1.

Hu, W., W. Li, and A. Abubakar, 2019, Machine learning and data analytics for geoscience applications: Geophysics.

Hu, S., W. Zhao, Z. Xu, H. Zeng, Q. Fu, L. Jiang, S. Shi, Z. Wang, and W. Liu, 2017, Applying principal component analysis to seismic attributes for interpretation of evaporite facies: Lower Triassic Jialingjiang Formation, Sichuan Basin, China: Interpretation, 5, T461–T475, doi:10.1190/INT-2017-0004.1.

Huang, G., Z. Liu, and K. Q. Weinberger, 2016, Densely Connected Convolutional Networks: CoRR, abs/1608.0.

Infante-Paez, L., and K. J. Marfurt, 2017, Seismic expression and geomorphology of igneous bodies: A Taranaki Basin, New Zealand, case study: Interpretation, 5, SK121–SK140, doi:10.1190/INT-2016-0244.1.

Jayaram, V., A. Roy, B. Barna, D. Devegowda, J. Floyd, P. Ashok, A. Abubakar, A. Kaul, and E. Schnetzler, 2019, Introduction to special section: Insights to digital oilfield data using artificial intelligence and big data analytics: Interpretation, 7, SFi-SFi, doi:10.1190/INT-2019-0618-SPSEINTRO.1.

Kim, Y., R. Hardisty, and K. J. Marfurt, 2019, Attribute selection in seismic facies classification: Application to a Gulf of Mexico 3D seismic survey and the Barnett Shale: Interpretation, 7, SE281–SE297, doi:10.1190/INT-2018-0246.1.

Kim, Y., and N. Nakata, 2018, Geophysical inversion versus machine learning in inverse problems: The Leading Edge, 37, 894–901, doi:10.1190/tle37120894.1.

Kingma, D. P., and J. Ba, 2014, Adam: A Method for Stochastic Optimization: arXiv e-prints, arXiv:1412.6980.

Kleinberg, J. M., 2003, An Impossibility Theorem for Clustering, in S. Becker, S. Thrun, and K. Obermayer, eds., Advances in Neural Information Processing Systems 15: MIT Press, 463–470.

Koch, G. R., 2015, Siamese Neural Networks for One-Shot Image Recognition, in 32nd International Conference on Machine Learning: JMLR.

Kohonen, T., 1990, The self-organizing map: Proceedings of the IEEE, 78, 1464–1480, doi:10.1109/5.58325.

Krizhevsky, A., I. Sutskever, and G. E. Hinton, 2012, ImageNet Classification with Deep Convolutional Neural Networks, in Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1: Curran Associates Inc., NIPS'12, 1097–1105.

La Marca-Molina, K., Silver, C., Bedle, H., Slatt, R., 2019. Seismic facies identification in a deepwater channel complex applying seismic attributes and unsupervised machine learning techniques. A case study in the Taranaki Basin, New Zealand., in: SEG Technical Program Expanded Abstracts 2019, SEG Technical Program Expanded Abstracts. Society of Exploration Geophysicists, pp. 2059–2063. https://doi.org/doi:10.1190/segam2019-3216705.1

LeCun, Y., 1998, The MNIST database of handwritten digits: http://yann. lecun. com/exdb/mnist/.

LeCun, Y., Y. Bengio, and G. Hinton, 2015, Deep learning: Nature, 521, 436–444, doi:10.1038/nature14539.

Li, H., W. Yang, and X. Yong, 2018, Deep learning for ground-roll noise attenuation, in SEG Technical Program Expanded Abstracts 2018: Society of Exploration Geophysicists, SEG Technical Program Expanded Abstracts, 1981–1985, doi:10.1190/segam2018-2981295.1.

Lim, H. H., A. A. Swartz, H. A. Yueh, J. A. Kong, R. T. Shin, and J. J. van Zyl, 1989, Classification of Earth terrain using polarimetric synthetic aperture radar images: Journal of Geophysical Research: Solid Earth, 94, 7049–7057, doi:10.1029/JB094iB06p07049.

Lubo-Robles, D., T. Ha, S. Lakshmivarahan, and K. J. Marfurt, 2019, Supervised seismic facies classification using probabilistic neural networks: Which attributes should the interpreter use?, in SEG Technical Program Expanded Abstracts 2019: Society of Exploration Geophysicists, SEG Technical Program Expanded Abstracts, 2273–2277, doi:10.1190/segam2019-3216841.1.

Lubo-Robles, D., and K. J. Marfurt, 2019, Independent Component Analysis for reservoir geomorphology and unsupervised seismic facies classification in the Taranaki Basin, New Zealand.: Interpretation, 1–76, doi:10.1190/int-2018-0109.1.

de Matos, M. C., P. L. Osorio, and P. R. Johann, 2007, Unsupervised seismic facies analysis using wavelet transform and self-organizing maps: Geophysics, 72, P9–P21, doi:10.1190/1.2392789.

de Matos, M. C., M. (Moe) Yenugu, S. M. Angelo, and K. J. Marfurt, 2011, Integrated seismic texture segmentation and cluster analysis applied to channel delineation and chert reservoir characterization: Geophysics, 76, P11–P21, doi:10.1190/geo2010-0150.1.

Morley, C. K., 2018, 3-D seismic imaging of the plumbing system of the Kora Volcano, Taranaki Basin, New Zealand: The influence of syn-rift structure on shallow igneous intrusion architecture: Geosphere, 14, 2533–2584, doi:10.1130/GES01645.1.

Mustafa, A., M. Alfarraj, and G. AlRegib, 2019, Estimation of acoustic impedance from seismic data using temporal convolutional network, in SEG Technical Program Expanded Abstracts 2019: Society of Exploration Geophysicists, SEG Technical Program Expanded Abstracts, 2554–2558, doi:10.1190/segam2019-3216840.1.

Olah, C., A. Mordvintsev, and L. Schubert, 2017, Feature Visualization: Distill, doi:10.23915/distill.00007.

Olah, C., A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, 2018, The Building Blocks of Interpretability: Distill, doi:10.23915/distill.00010.

Palacio-Niño, J.-O., and F. Berzal, 2019, Evaluation Metrics for Unsupervised Learning Algorithms.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, 2011, Scikit-learn: Machine Learning in Python: Journal of Machine Learning Research, 12, 2825–2830.

Perol, T., M. Gharbi, and M. Denolle, 2018, Convolutional neural network for earthquake detection and location: Science Advances, 4, e1700578, doi:10.1126/sciadv.1700578.

Pham, N., S. Fomel, and D. Dunlap, 2019, Automatic channel detection using deep learning: Interpretation, 7, SE43–SE50, doi:10.1190/INT-2018-0202.1.

Pires de Lima, R., A. Bonar, D. D. Coronado, K. Marfurt, and C. Nicholson, 2019, Deep convolutional neural networks as a geological image classification tool: The Sedimentary Record, 17, 4–9, doi:10.210/sedred.2019.2.

Pires de Lima, R., and K. J. Marfurt, 2018, Principal component analysis and K-means analysis of airborne gamma-ray spectrometry surveys: doi:10.1190/segam2018-2996506.1.

Pires de Lima, R., F. Suriamin, K. J. Marfurt, and M. J. Pranter, 2019, Convolutional neural networks as aid in core lithofacies classification: Interpretation, 7, SF27–SF40, doi:10.1190/INT-2018-0245.1.

Qi, J., T. Lin, T. Zhao, F. Li, and K. Marfurt, 2016, Semisupervised multiattribute seismic facies analysis: Interpretation, 4, SB91–SB106, doi:10.1190/INT-2015-0098.1.

Ran, X., L. Xue, Y. Zhang, Z. Liu, X. Sang, and J. He, 2019, Rock Classification from Field Image Patches Analyzed Using a Deep Convolutional Neural Network: Mathematics, 7, 755, doi:10.3390/math7080755.

Ren, C. X., O. Dorostkar, B. Rouet-Leduc, C. Hulbert, D. Strebel, R. A. Guyer, P. A. Johnson, and J. Carmeliet, 2019, Machine Learning Reveals the State of Intermittent Frictional Dynamics in a Sheared Granular Fault: Geophysical Research Letters, 46, 7395–7403, doi:10.1029/2019GL082706.

Roden, R., T. Smith, and D. Sacrey, 2015, Geologic pattern recognition from seismic attributes: Principal component analysis and self-organizing maps: Interpretation, 3, SAE59–SAE83, doi:10.1190/INT-2015-0037.1.

Ronneberger, O., P. Fischer, and T. Brox, 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation, in N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds., Medical Image Computing and Computer-Assisted Intervention -- MICCAI 2015: Springer International Publishing, 234–241.

Rosenblatt, F., 1957, The Perceptron — A Perceiving and Recognizing Automaton: Tech. Rep. 85-460-1 (Cornell Aeronautical Laboratory).

Rostami, M., S. Kolouri, E. Eaton, and K. Kim, 2019, Deep Transfer Learning for Few-Shot SAR Image Classification: Remote Sensing, 11, 1374, doi:10.3390/rs11111374.

Roy, A., A. S. Romero-Peláez, T. J. Kwiatkowski, and K. J. Marfurt, 2014, Generative topographic mapping for seismic facies estimation of a carbonate wash, Veracruz Basin, southern Mexico: Interpretation, 2, SA31–SA47, doi:10.1190/INT-2013-0077.1.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, 2015, ImageNet Large Scale Visual Recognition Challenge: International Journal of Computer Vision, 115, 211–252, doi:10.1007/s11263-015-0816-y.

Russell, B., 2019, Machine learning and geophysical inversion — A numerical study: The Leading Edge, 38, 512–519, doi:10.1190/tle38070512.1.

Russell, B. H., 2004, The application of multivariate statistics and neural networks to the prediction of reservoir parameters using seismic attributes: University of Calgary (Canada).

Schroff, F., D. Kalenichenko, and J. Philbin, 2015, FaceNet: A Unified Embedding for Face Recognition and Clustering, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Selfridge, O. G., 1958, Pandemonium: a paradigm for learning in mechanisation of thought processes, in Proceedings of Symposium on Mechanisation of Thought Processes: 513–526.

Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, 2017, Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization, in 2017 IEEE International Conference on Computer Vision (ICCV): 618–626, doi:10.1109/ICCV.2017.74.

Shaw, S., A. Sharma, R. Baraniuk, and B. Roy, 2019, Introduction to this special section: Machine learning applications: The Leading Edge, 38, 510, doi:10.1190/tle38070510.1.

Shi, Y., X. Wu, and S. Fomel, 2019, SaltSeg: Automatic 3D salt segmentation using a deep convolutional neural network: Interpretation, 7, SE113–SE122, doi:10.1190/INT-2018-0235.1.

Simonyan, K., A. Vedaldi, and A. Zisserman, 2013, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps: CoRR, abs/1312.6.

Simonyan, K., and A. Zisserman, 2014, Very Deep Convolutional Networks for Large-Scale Image Recognition: ArXiv e–prints.

Sinha, S., Y. Wen, R. A. Pires de Lima, and K. Marfurt, 2018, Statistical controls on induced seismicity: Unconventional Resources Technology Conference, doi:10.15530/urtec-2018-2897507-MS.

Springenberg, J. T., A. Dosovitskiy, T. Brox, and M. Riedmiller, 2014, Striving for Simplicity: The All Convolutional Net: arXiv e-prints, arXiv:1412.6806.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 2014, Dropout: A Simple Way to Prevent Neural Networks from Overfitting: Journal of Machine Learning Research, 15, 1929–1958.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, 2014, Going Deeper with Convolutions: CoRR, abs/1409.4.

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, 2015, Rethinking the Inception Architecture for Computer Vision: arXiv e-prints, arXiv:1512.00567.

Valentín, M. B., C. R. Bom, J. M. Coelho, M. D. Correia, Márcio P. de Albuquerque, Marcelo P. de Albuquerque, and E. L. Faria, 2019, A deep residual convolutional neural network for automatic lithological facies identification of Brazilian pre-salt oilfield wellbore image logs: Journal of Petroleum Science and Engineering, doi:10.1016/J.PETROL.2019.04.030.

Waddell, D. E., 1966, Pennsylvanian fusulinids in the Ardmore Basin - Love and Carter counties, Oklahoma: Oklahoma Geological Survey Bulletin 113, 128 p.

Waldeland, A. U., A. C. Jensen, L.-J. Gelius, and A. H. S. Solberg, 2018, Convolutional neural networks for automated seismic interpretation: The Leading Edge, 37, 529–537, doi:10.1190/tle37070529.1.

Wang, Y., C. H. Arns, S. S. Rahman, and J.-Y. Arns, 2018, Porous Structure Reconstruction Using Convolutional Neural Networks: Mathematical Geosciences, 50, 781–799, doi:10.1007/s11004-018-9743-0.

Wang, B., N. Zhang, W. Lu, and J. Wang, 2019, Deep-learning-based seismic data interpolation: A preliminary result: GEOPHYSICS, 84, V11–V20, doi:10.1190/geo2017-0495.1.

Wickstrøm, K., M. Kampffmeyer, and R. Jenssen, 2018, Uncertainty modeling and interpretability in convolutional neural networks for polyp segmentation, in 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP): 1–6, doi:10.1109/MLSP.2018.8516998.

Wu, X., 2017, Directional structure-tensor-based coherence to detect seismic faults and channels: GEOPHYSICS, 82, A13--A17, doi:10.1190/geo2016-0473.1.

Wu, X., L. Liang, Y. Shi, and S. Fomel, 2019, FaultSeg3D: using synthetic datasets to train an end-to-end convolutional neural network for 3D seismic fault segmentation: GEOPHYSICS, 1–36, doi:10.1190/geo2018-0646.1.

Yang, F., and J. Ma, 2019, Deep-learning inversion: A next-generation seismic velocity model building method: GEOPHYSICS, 84, R583–R599, doi:10.1190/geo2018-0249.1.

Zech, J. R., M. A. Badgeley, M. Liu, A. B. Costa, J. J. Titano, and E. K. Oermann, 2018, Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study: PLOS Medicine, 15, e1002683, doi:10.1371/journal.pmed.1002683.

Zhao, T., 2018, Seismic facies classification using different deep convolutional neural networks, in SEG Technical Program Expanded Abstracts 2018: Society of Exploration Geophysicists, 2046–2050, doi:10.1190/segam2018-2997085.1.

Zhao, T., V. Jayaram, A. Roy, and K. J. Marfurt, 2015, A comparison of classification techniques for seismic facies recognition: Interpretation, 3, SAE29–SAE58, doi:10.1190/INT-2015-0044.1.

Zhao, T., F. Li, and K. J. Marfurt, 2018, Seismic attribute selection for unsupervised seismic facies analysis using user-guided data-adaptive weights: GEOPHYSICS, 83, O31--O44, doi:10.1190/geo2017-0192.1.

Zhao, Y., Y. Li, and B. Yang, 2019, Low-Frequency Desert Noise Intelligent Suppression in Seismic Data Based on Multiscale Geometric Analysis Convolutional Neural Network: IEEE Transactions on Geoscience and Remote Sensing, 1–16, doi:10.1109/TGRS.2019.2938836.

Zhao, T., J. Zhang, F. Li, and K. J. Marfurt, 2016, Characterizing a turbidite system in Canterbury Basin, New Zealand, using seismic attributes and distance-preserving self-organizing maps: Interpretation, 4, SB79–SB89, doi:10.1190/INT-2015-0094.1.

Zhou, B., A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, 2016, Learning Deep Features for Discriminative Localization, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR): 2921–2929, doi:10.1109/CVPR.2016.319.

Zhou, W., S. Newsam, C. Li, and Z. Shao, 2018, PatternNet: A benchmark dataset for performance evaluation of remote sensing image retrieval: ISPRS Journal of Photogrammetry and Remote Sensing, 145, 197–209, doi:10.1016/J.ISPRSJPRS.2018.01.004.

Chapter 2: Deep convolutional neural networks as a geological image classification tool

Rafael Pires de Lima[1,2], Alicia Bonar[1], David Duarte Coronado[1], Kurt Marfurt[1], Charles Nicholson[3]

[1]School of Geology and Geophysics, The University of Oklahoma, 100 East Boyd Street, RM 710, Norman, Oklahoma, 73019, USA

[2]The Geological Survey of Brazil – CPRM, 55 Rua Costa, São Paulo, São Paulo, Brazil

[3]School of Industrial and Systems Engineering, The University of Oklahoma, 202 West Boyd Street, RM 124, Norman, Oklahoma, 73019, USA

Preface

This chapter is presented as it was published in The Sedimentary Record (Pires de Lima et al., 2019a), which itself was based on an EAGE expanded abstract (Pires de Lima et al., 2019b). This chapter shows the results of the application of transfer learning to different sets of geoscience images, without delving into the details and challenges encountered in each one of the tasks. The application presented here shows the potential of the use of convolutional neural networks for different fields of geosciences. This chapter is also presented in my Data Science and Analytics Master's Thesis.

References:

Pires de Lima, R., Bonar, A., Coronado, D.D., Marfurt, K., Nicholson, C., 2019a. Deep convolutional neural networks as a geological image classification tool. Sediment. Rec. 17, 4–9. https://doi.org/10.210/sedred.2019.2

Pires de Lima, R., Marfurt, K., Duarte, D., Bonar, A., 2019b. Progress and Challenges in Deep Learning Analysis of Geoscience Images, in: 81st EAGE Conference and Exhibition 2019. EAGE. https://doi.org/10.3997/2214-4609.201901607

Abstract

A convolutional neural network (CNN) is a deep learning (DL) method that has been widely and successfully applied to computer vision tasks including object localization, detection, and image classification. DL for supervised learning tasks is a method that uses the raw data to determine the classification features, in contrast to other machine learning (ML) techniques that require pre-selection of the input features (or attributes). In the geosciences, we hypothesize that deep learning will facilitate the analysis of uninterpreted images that have been neglected due to a limited number of experts, such as fossil images, slabbed cores, or petrographic thin sections. We use transfer learning, which employs previously trained models to shorten the development time for subsequent models, to address a suite of geologic interpretation tasks that may benefit from ML. Using two different base models, MobileNet V2 and Inception V3, we illustrate the successful classification of microfossils, core images, petrographic photomicrographs, and rock and mineral hand sample images. ML does not replace the expert geoscientist. The expert defines the labels (interpretations) needed to train the algorithm and also monitors the results to address incorrect or ambiguous classifications. ML techniques provide a means to apply the expertise of skilled geoscientists to much larger volumes of data

Introduction

Machine learning (ML) techniques have been successfully applied, with considerable success, in the geosciences for almost two decades. Applications of ML by the geoscientific community include many examples such as seismic-facies classification (Meldahl et al., 2001; West et al., 2002; de Matos et al., 2011; Roy et al., 2014; Qi et al., 2016; Hu et al., 2017; Zhao et al., 2017), electrofacies classification (Allen and Pranter, 2016), and analysis of seismicity

(Kortström et al., 2016; DeVries et al., 2018; Perol et al., 2018; Sinha et al., 2018), and classification of volcanic ash (Shoji et al., 2018), among others. Conventionally, ML applications rely on a set of attributes (or features) selected or designed by an expert. Features are specific characteristics of an object that can be used to study patterns or predict outcomes. In classification modeling, these features are chosen with the goal of distinguishing one object from another.

Typically, feature selection is problem dependent. For example, a clastic sedimentary rock is most broadly classified by its grain size; therefore, a general classification for a rock sample (data) is sandstone if its grain sizes (features) lie from 0.06 mm to 2.0 mm following the Wentworth size class. In this example, a single feature is used to classify the sample, but more complex and/or detailed classification often requires analysis of multiple features exhibited by the sample. An inefficiency of traditional ML approaches is that many features may be constructed while only a subset of them are actually needed for the classification.

The use of explicitly designed features to classify data was the traditional approach in ML applications within the geosciences as in many other research areas. This classification approach works well when human interpreters know and can quantify the features that distinguish one object from another. However, sometimes an interpreter will subconsciously classify features and have difficulty describing what the distinguishing features might be, relying on "I'll know what the object is when I see it". In contrast to feature-driven ML classification algorithms, deep learning (DL) models extract information directly from the raw unstructured data rather than the data being manually transformed.

Because of their greater complexity (and resulting flexibility and power) convolutional neural networks (CNN) usually requires more training data than traditional ML processes.

However, when expert-labeled data are provided, non-experts can use the CNN models to generate highly accurate results (e.g. TGS Salt Identification Challenge | Kaggle, 2019).

DL applications in the geosciences require experts to first define the labels used to construct the necessary data sets as well as identify and address any ambiguous results and anomalies. In order to bring awareness and provide basic information regarding CNN models, DL techniques, and the necessity of expert-level knowledge needed to utilize these advancements, we applied these methods to four different geologic tasks. Figure 1 shows samples of different types of data that can be interpreted and labeled by experienced geologists. We use such interpretations to train our models. In this manuscript, we show how CNN can aid geoscientists with microfossil identification, core descriptions, petrographic analyses, and as a potential tool for education and outreach by creating a simple hand specimen identification application.

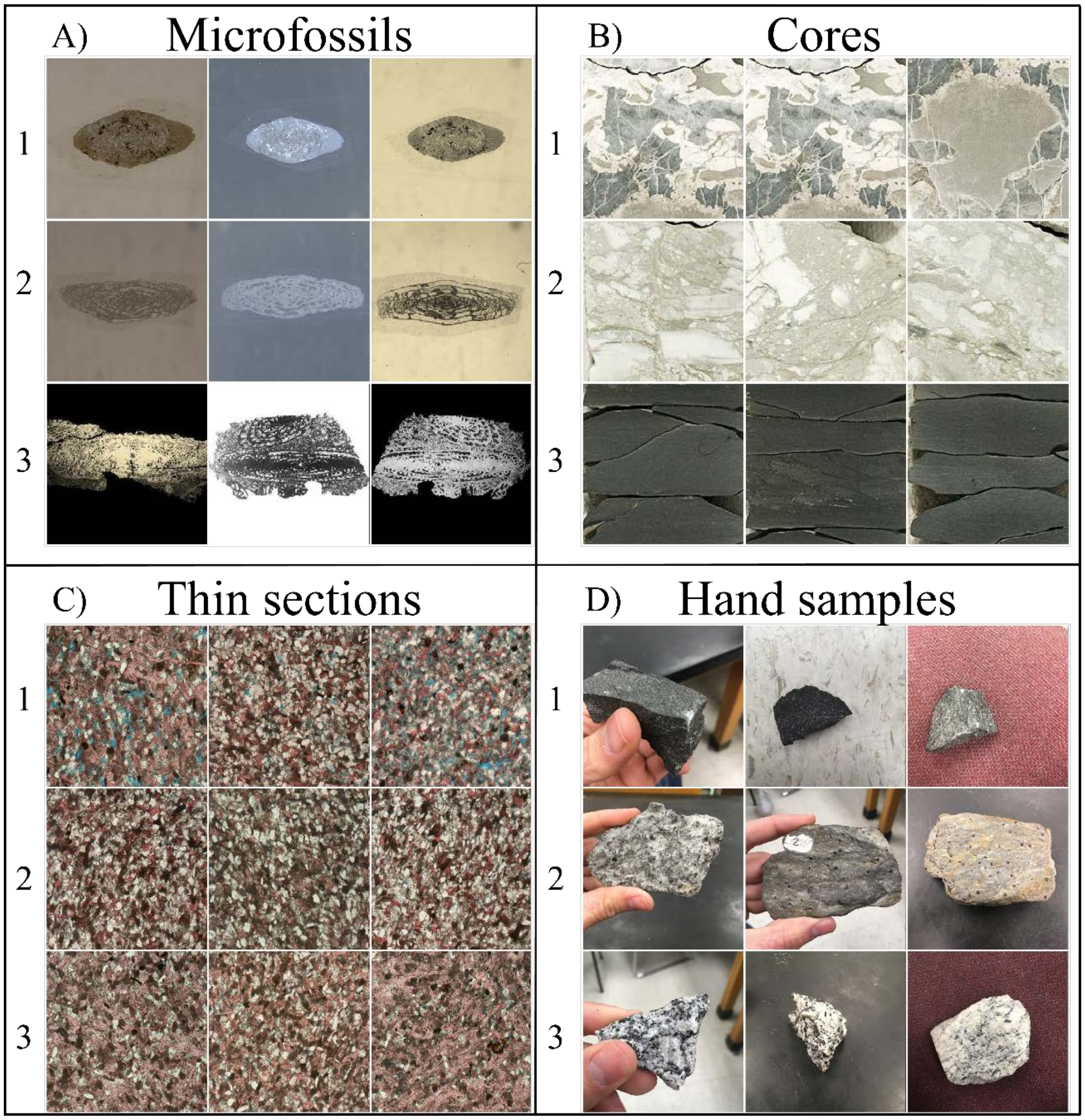


Figure 15: Examples of the data used in this study. A) Three of the seven Fusulinids groups (*Beedeina* (1), *Fusulinella* (2), and *Parafusulina* (3)). B) Three of the five lithofacies (bioturbated mudstone-wackestone (1), chert breccia (2), and shale (3)). C) Reservoir quality classes (high (1), intermediate (2), and low (3)) D) Three of the six rock sample groups (basalt (1), garnet schist (2), and granite (3)). Samples were interpreted by professionals working with each separate dataset.

Convolutional neural networks and transfer learning

Recent CNN research has yielded significant improvements and unprecedented accuracy (the ratio between correct classifications and the total number of samples classified) in image classification and are recognized as leading methods for large-scale visual recognition problems, such as the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC, Russakovsky et al. (2015)). Specific CNN architectures have been the leading approach for several years now (e.g., Szegedy et al., 2014; Chollet, 2016; He et al., 2016; Huang et al., 2016; Sandler et al., 2018). Researchers noted that the parameters learned by the layers in many CNN models trained on images exhibit a common behavior – layers closer to the input data tend to learn general features, such as edge detecting/enhancing filters or color blobs, then there is a transition to more specific dataset features, such as faces, feathers, or object parts (Yosinski et al., 2014; Yin et al., 2017). These general-specific CNN layer properties are important points to be considered for the implementation of transfer learning (Caruana, 1995; Bengio, 2012; Yosinski et al., 2014). In transfer learning, first a CNN model is trained on a base dataset for a specific task. The learned features (model parameters) are repurposed, or transferred, to a second target CNN to be trained on a different dataset and task (Yosinski et al., 2014).

New DL applications often require large volumes of data, however the combination of CNNs and transfer learning allows the reuse of existing DL models to novel classification problems with limited data, as has been demonstrated in diverse fields, such as botany (Carranza-Rojas et al., 2017), cancer classification (Esteva et al., 2017), and aircraft detection (Chen et al., 2018). Analyzing medical image data, Tajbakhsh et al. (2016) and Qayyum et al. (2017) found that transfer learning achieved comparable or better results than training a CNN model with randomly initialized parameters. As an example, training the entire InceptionV3 (Szegedy et al., 2015) with 1000 images (five classes, 50 original images for each class, four copies of each original image) with randomly initialized parameters can be 10 times slower than the transfer learning process (11 minutes vs 1 minute on average for five executions) using a Nvidia Quadro

M2000 (768 CUDA Cores). On a CPU (3.60 GHz clock speed), training the entire model can take up to 2 hours whereas transfer learning can be completed within a few minutes. We also noticed that transfer learning is easier to train. During the speed comparison test, transfer learning achieved high accuracies (close to 1.0) within 5 epochs (note the dataset is very simple with most of the samples being copies of each other). Successful applications of computer vision technologies in different fields suggest that ML models could be extremely beneficial for geologic applications, especially those in the category of image classification problems.

For the examples we present in this paper (Figure 1), we rely on the use of transfer learning (Yosinski et al., 2014) using the MobileNetV2 (Sandler et al., 2018) and InceptionV3 as our base CNN models. Both MobileNetV2 and InceptionV3 were trained on ILSVRC. Therefore, the CNN models we used were constructed based on inputs of 3-channels (RGB) of 2D photographic images. We randomly select part of the data to be used as a test set maintaining the same proportion of samples per class as in the training set. The data in the test set is not used during the computational process for model training; rather, it is used to evaluate the quality and robustness of the final model. Due to limited space, we refrained showing the CNN mistakes and many of the steps necessary for data preparation.

CNN-Assisted fossil analysis

Biostratigraphy has become a less common focus of study in the discipline of paleontology (Farley and Armentrout, 2000, 2002), but the applications of biostratigraphy are necessary for understanding age-constraints for rocks that cannot be radiometrically dated. Access to a specific taxonomic expert to accurately analyze fossils at the species-level can be as challenging as data acquisition and preparation. Using labeled data from the University of

Oklahoma Sam Noble Museum and iDigBio portal, we found that Fusulinids (index fossils for the Late Paleozoic) can be accurately classified with the use of transfer learning. Accurate identification of a Fusulinid depends on characteristics that must be observed and exposed along the long axis of the (prolate spheroid-shaped) Fusulinid. We used a dataset of 1850 qualified images including seven different Fusulinid genera. After retraining the CNN model, we obtained an accuracy for the test set (10% of the data) of 1.0 for both retrained MobileNetV2 and InceptionV3 (Table 1). Figure 2 shows a schematic view of the classification process.

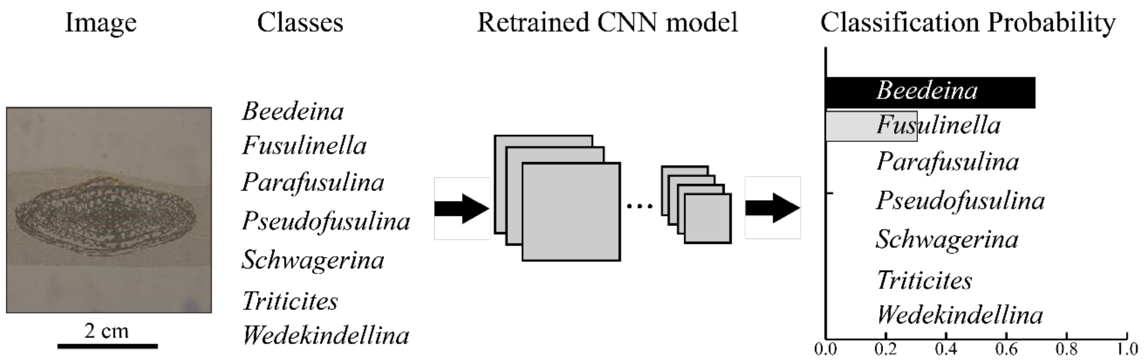


Figure 16: An example of the classification process. In this example, a thin-section image that should fit one of the seven Fusulinid genera is analyzed by the model. The model outputs the probability assigned to each of the possible classes (all probabilities summing to 1.0). The term “classes” here is used in the ML sense rather than the biological one. In the example provided, our model provided a high probability for the same class as the human expert. Note that in the implementation we use the model will classify any image as one of the seven learned classes – even if the image is clearly not a fossil. This highlights the importance of a domain expert intervention.

Table 1: Summary of test accuracy for the examples in this study.

Dataset	**Number of training samples**	**Number of test samples**	**Number of output classes**	**MobileNetV2 Accuracy**	**InceptionV3 Accuracy**
Microfossils (Fusulinids)	1480	184	7	1.00	1.00
Core	227	28	5	1.00	0.97
Petrographic thin-sections	194	31	3	0.81	0.81
Rock samples	1218	151	6	0.98	0.97

CNN-Assisted core description

Miles of drilled cores are stored in boxes in enormous warehouses, many of which have either been neglected for years or never digitally described. Core-based rock-type descriptions are important for understanding the lithology and structure of subsurface geology. Using several hundred feet of labeled core from a Mississippian limestone in Oklahoma (data from Suriamin and Pranter, 2018 and Pires de Lima et al., 2019), we selected a small sample of 285 images from five distinct lithofacies to be classified by the retrained CNN models. Pires de Lima et al. (2019) describes how a sliding window is used to generate CNN input data, cropping small sections from a standard core image. We used 10% of the data as the test set and achieved an accuracy of 1.0 using the retrained MobileNetV2 and an accuracy of 0.97 using the retrained InceptionV3 (Table 1).

CNN-Assisted reservoir quality classification using petrographic thin sections

Petrography focuses on the microscopic description and classification of rocks and is one of the most important techniques in sedimentary and diagenetic studies. Potential information gained from thin section analysis compared to hand specimen descriptions include mineral distribution and percentage, pore space analysis, and cement composition. Petrographic analyses can be laborious even for experienced geologists. Using a total of 161 photomicrographs of parallel Nicol polarization of thin sections from the Sycamore Formation shale resource play in Oklahoma, we classified these images as representatives of high, intermediate, and low reservoir quality depending on the percent of calcite cement and pore space. We used 20% of the images in the test set and obtained a test set accuracy of 0.81 for both the retrained MobileNetV2 and the retrained InceptionV3 (Table 1).

CNN-Assisted rock sample analysis

By creating a simple website, the general population could have immediate access to a rock identification tool using transfer learning technology. For this work in progress, we used smartphones to acquire 1521 pictures of six different rock types, using five different hand samples for each one of the rock types. We took pictures with different backgrounds, as visually depicted in Figure 1, however all pictures were taken in the same classroom. After retraining the CNN models, we obtained an accuracy for the test set (10% of original data) of 0.98 using the retrained MobileNetV2 and 0.97 using the retrained InceptionV3 (Table 1). We note that our model does not perform well with no-background images (i.e., pictures in which the rock sample is edited and seems to be within a white or black canvas) as such images were not used in training.

Conclusions and future work

Although gaining popularity and becoming established as robust technologies in other scientific fields, transfer learning and CNN models are still novel with respect to application within the geoscience community. In this paper, we used CNN and transfer learning to address four potential applications that could improve data management, organization, and interpretation in different segments of our community. We predict that the versatile transfer learning and deep learning technologies will play a role in public education and community outreach, allowing the public to identify rock samples much as they currently can use smart phone apps to identify visitors to their bird feeder. Such public engagement will increase geological awareness and provide learning opportunities for elementary schools, outdoor organizations, and families.

For all of our examples, we were able to achieve high levels of accuracy (greater than 0.81) by repurposing two different CNN models originally assembled for generic computer vision tasks. We note that the examples and applications demonstrated here are curated, and therefore we expected highly accurate results. We presented demonstrations with limited classes and relatively well-controlled input images, so near perfect accuracies cannot necessarily be expected in an open, free-range deployment scenario. Regardless, the ability to create distinctive models for specific sets of images allows for a versatile application.

The techniques we have shown could greatly improve the speed of monotonous tasks such as describing miles of core data with very similar characteristics or looking at hundreds of thin sections from the same geologic formation. While the tasks are performed by the computer, the geoscience expert is still the most important element in every analysis in order to create the necessary datasets and provide quality control of the generated results. In the end, the expert validates the correctness of the results and looks for anomalies that are poorly represented by the target classes. We believe ML can help maintain consistency in interpretations and even provide a resource for less common observations and data variations, such as previously overlooked fossil subspecies and unique mineralogical assemblages in small communities and private collections, thereby building and reconciling a more complete international database. By combing expert knowledge and time efficient technology, ML methods can accelerate many data analysis processes for geologic research.

Acknowledgements

We thank the iDigBio initiative for providing access to the community for biodiversity collections data. Rafael acknowledges CNPq (grant 203589/2014-9) for the financial support and CPRM for granting the leave of absence allowing the pursuit of his Ph.D. studies. We thank Roger J. Burkhalter from the University of Oklahoma Sam Noble Museum of Natural History for providing the Fusulinids images used in this manuscript.

References

Allen, D.B., Pranter, M.J., 2016. Geologically constrained electrofacies classification of fluvial deposits: An example from the Cretaceous Mesaverde Group, Uinta and Piceance Basins. Am. Assoc. Pet. Geol. Bull. 100, 1775–1801. https://doi.org/10.1306/05131614229

Bengio, Y., 2012. Deep Learning of Representations for Unsupervised and Transfer Learning, in: Guyon, I., Dror, G., Lemaire, V., Taylor, G., Silver, D. (Eds.), Proceedings of ICML Workshop on Unsupervised and Transfer Learning, Proceedings of Machine Learning Research. PMLR, Bellevue, Washington, USA, pp. 17–36.

Carranza-Rojas, J., Goeau, H., Bonnet, P., Mata-Montero, E., Joly, A., 2017. Going deeper in the automated identification of Herbarium specimens. BMC Evol. Biol. 17, 181. https://doi.org/10.1186/s12862-017-1014-z

Caruana, R., 1995. Learning Many Related Tasks at the Same Time with Backpropagation, in: Tesauro, G., Touretzky, D.S., Leen, T.K. (Eds.), Advances in Neural Information Processing Systems 7. MIT Press, pp. 657–664.

Chen, Z., Zhang, T., Ouyang, C., Chen, Z., Zhang, T., Ouyang, C., 2018. End-to-End Airplane Detection Using Transfer Learning in Remote Sensing Images. Remote Sens. 10, 139. https://doi.org/10.3390/rs10010139

Chollet, F., 2016. Xception: Deep Learning with Depthwise Separable Convolutions. CoRR abs/1610.0.

de Matos, M.C., Yenugu, M. (Moe), Angelo, S.M., Marfurt, K.J., 2011. Integrated seismic texture segmentation and cluster analysis applied to channel delineation and chert reservoir characterization. Geophysics 76, P11–P21. https://doi.org/10.1190/geo2010-0150.1

DeVries, P.M.R., Viégas, F., Wattenberg, M., Meade, B.J., 2018. Deep learning of aftershock patterns following large earthquakes. Nature 560, 632–634. https://doi.org/10.1038/s41586-018-0438-y

Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S., 2017. Dermatologist-level classification of skin cancer with deep neural networks. Nature 542, 115–118. https://doi.org/10.1038/nature21056

Farley, M.B., Armentrout, J.M., 2002. Tools, Biostratigraphy becoming lost art in rush to find new exploration. Offshore 94–95.

Farley, M.B., Armentrout, J.M., 2000. Fossils in the Oil Patch. Geotimes 14–17.
He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep Residual Learning for Image Recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90
Hu, S., Zhao, W., Xu, Z., Zeng, H., Fu, Q., Jiang, L., Shi, S., Wang, Z., Liu, W., 2017. Applying principal component analysis to seismic attributes for interpretation of evaporite facies: Lower Triassic Jialingjiang Formation, Sichuan Basin, China. Interpretation 5, T461–T475. https://doi.org/10.1190/INT-2017-0004.1
Huang, G., Liu, Z., Weinberger, K.Q., 2016. Densely Connected Convolutional Networks. CoRR abs/1608.0.
Kortström, J., Uski, M., Tiira, T., 2016. Automatic classification of seismic events within a regional seismograph network. Comput. Geosci. 87, 22–30. https://doi.org/10.1016/J.CAGEO.2015.11.006
Meldahl, P., Heggland, R., Bril, B., de Groot, P., 2001. Identifying faults and gas chimneys using multiattributes and neural networks. Lead. Edge 20, 474–482. https://doi.org/10.1190/1.1438976
Perol, T., Gharbi, M., Denolle, M., 2018. Convolutional neural network for earthquake detection and location. Sci. Adv. 4, e1700578. https://doi.org/10.1126/sciadv.1700578
Pires de Lima, R., Suriamin, F., Marfurt, K.J., Pranter, M.J., 2019. Convolutional neural networks as aid in core lithofacies classification. Interpretation 7, SF27–SF40. https://doi.org/10.1190/INT-2018-0245.1
Qayyum, A., Anwar, S.M., Awais, M., Majid, M., 2017. Medical image retrieval using deep convolutional neural network. Neurocomputing 266, 8–20. https://doi.org/10.1016/J.NEUCOM.2017.05.025
Qi, J., Lin, T., Zhao, T., Li, F., Marfurt, K., 2016. Semisupervised multiattribute seismic facies analysis. Interpretation 4, SB91–SB106. https://doi.org/10.1190/INT-2015-0098.1
Roy, A., Romero-Peláez, A.S., Kwiatkowski, T.J., Marfurt, K.J., 2014. Generative topographic mapping for seismic facies estimation of a carbonate wash, Veracruz Basin, southern Mexico. Interpretation 2, SA31–SA47. https://doi.org/10.1190/INT-2013-0077.1
Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vis. 115, 211–252. https://doi.org/10.1007/s11263-015-0816-y
Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. ArXiv e-prints.
Shoji, D., Noguchi, R., Otsuki, S., Hino, H., 2018. Classification of volcanic ash particles using a convolutional neural network and probability. Sci. Rep. 8, 8111. https://doi.org/10.1038/s41598-018-26200-2
Sinha, S., Wen, Y., Pires de Lima, R.A., Marfurt, K., 2018. Statistical controls on induced seismicity. Unconventional Resources Technology Conference. https://doi.org/10.15530/urtec-2018-2897507-MS
Suriamin, F., Pranter, M.J., 2018. Stratigraphic and lithofacies control on pore characteristics of Mississippian limestone and chert reservoirs of north-central Oklahoma. Interpretation 1–66. https://doi.org/10.1190/int-2017-0204.1
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2014. Going Deeper with Convolutions. CoRR abs/1409.4.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2015. Rethinking the Inception Architecture for Computer Vision. CoRR abs/1512.0.

Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B., Liang, J., 2016. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? IEEE Trans. Med. Imaging 35, 1299–1312. https://doi.org/10.1109/TMI.2016.2535302

TGS Salt Identification Challenge | Kaggle [WWW Document], n.d. URL https://www.kaggle.com/c/tgs-salt-identification-challenge (accessed 1.10.19).

West, B.P., May, S.R., Eastwood, J.E., Rossen, C., 2002. Interactive seismic facies classification using textural attributes and neural networks. Lead. Edge 21, 1042–1049. https://doi.org/10.1190/1.1518444

Yin, X., Chen, W., Wu, X., Yue, H., 2017. Fine-tuning and visualization of convolutional neural networks, in: 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA). IEEE, pp. 1310–1315. https://doi.org/10.1109/ICIEA.2017.8283041

Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks? Adv. Neural Inf. Process. Syst. 27, 3320–3328.

Zhao, T., Li, F., Marfurt, K.J., 2017. Constraining self-organizing map facies analysis with stratigraphy: An approach to increase the credibility in automatic seismic facies classification. Interpretation 5, T163–T171. https://doi.org/10.1190/INT-2016-0132.1

Chapter 3: Convolutional neural networks as aid in core lithofacies classification

Rafael Pires de Lima[1,2], Fnu Suriamin[1,3], Kurt J. Marfurt[1], Matthew J. Pranter[1]

[1]School of Geology and Geophysics, The University of Oklahoma, 100 East Boyd Street, RM 710, Norman, Oklahoma, 73019, USA

[2]The Geological Survey of Brazil – CPRM, 55 Rua Costa, São Paulo, São Paulo, Brazil

[3]Oklahoma Geological Survey, 100 East Boyd Street, Room N-131, Norman, Oklahoma, 73019, USA

Preface

This chapter is presented as it was published in Interpretation (Pires de Lima et al., 2019). A preliminary piece was published in AAPG's Explorer (Pires de Lima et al., 2018). This chapter shows the results of the application of transfer learning to classify lithofacies from a core from the Mississippian limestone and chert reservoirs in the Anadarko Shelf, Grant County, Oklahoma.

References:

Pires de Lima, R., Marfurt, K., Suriamin, F., Pranter, M., Soreghan, G., 2018. Convolutional Neural Networks – If they can identify an oncoming car, can they identify lithofacies in core? AAPG Explorer.

Pires de Lima, R., Marfurt, K., Duarte, D., Bonar, A., 2019b. Progress and Challenges in Deep Learning Analysis of Geoscience Images, in: 81st EAGE Conference and Exhibition 2019. EAGE. https://doi.org/10.3997/2214-4609.201901607

Abstract

Artificial intelligence methods have a very wide range of applications. From speech recognition to self-driving cars, the development of modern deep learning architectures is helping researchers achieve new levels of accuracy in different fields. Although deep convolutional neural networks (a kind of deep learning technique) have reached or surpassed human-level performance in image recognition tasks, little has been done to transport this new image classification technology to geoscientific problems. We present what we believe to be the first paper using convolutional neural networks to identify lithofacies in cores. We use highly accurate models (trained with millions of images) and transfer learning to classify images of cored carbonate rocks. We show that different modern convolutional neural network architectures can achieve high levels of lithological image classification accuracy (~90%) and can be used to aid in the core description task. This core image classification technique has the potential to greatly standardize and accelerate the description process. We also provide the community with a new set of labeled data that can be used for further geologic/data science studies.

Introduction

Advances in deep learning and artificial intelligence promise to not only drive our cars but also taste our beer (Daily et al., 2017; Gardner et al., 1994). Specifically, recent advances in the architecture of deep learning convolutional neural networks (CNN) have brought the field of image classification and computer vision to a new level. Very deep convolutional neural networks emerged in 2014 and have achieved new levels of accuracy in several artificial intelligence classification problems (Szegedy et al., 2014). The current benchmark in object category classification and detection, named ImageNet, consists of hundreds of mixed-object categories and millions of images (Deng et al., 2009; Russakovsky et al., 2015) and it is commonly used to train CNNs. Current CNN models are able to differentiate the image of a leopard from that of a container ship, but moreover can differentiate images of leopards from their biological cousins -cheetahs and snow leopards (Krizhevsky et al., 2012).

Although machine learning has been significantly used in geoscience fields, the application of this technique in core-based lithofacies identification, a key component to better understand oil and gas reservoirs, is still limited. Machine learning techniques have been intensely used to aid seismic-facies classification (de Matos et al., 2011, 2007; Qi et al., 2016; Qian et al., 2018; Roy et al., 2014; Zhao et al., 2017, 2016), electrofacies classification (Allen and Pranter, 2016); lithofacies classification from well logs (Baldwin et al., 1990; Bestagini et al., 2017; Zhang et al., 1999), to predict permeability in tight sands (Zhang et al., 2018), and even for seismicity studies (Kortström et al., 2016; Perol et al., 2018; Sinha et al., 2018; Wu et al., 2018). Cored wells are important as they are the only data that provide the ground-truth of subsurface reservoirs including the lithofacies variations. The goals of core-based rock-type descriptions are to identify key lithofacies and facies associations, evaluate facies stacking and identify stratigraphic surfaces, interpret depositional environments, evaluate relationships among porosity, permeability, and lithofacies, and help operators to identify optimal zones for designing completions. Traditional core-based lithofacies identification is challenging as it is costly, time consuming, and subjective (e.g. different geologists describe the same core may yield different results). To address some of the core-based lithofacies identification challenges, we evaluate whether a CNN can help a specialist on their image-recognition task.

CNN goes hand-in-hand with the construction and archival of digital data bases. Many museums are now busy digitizing and sharing their collections (Blagoderov et al., 2012; Ellwood et al., 2015) With the exception of core measured by deep sea drilling projects and the like (e.g. NOAA, 2016), core images are not readily available. As an example, more than 100 miles of cores are stored in the Oklahoma Petroleum Information Center (OPIC), managed by the Oklahoma Geologic Survey. Other states and countries have similar repositories (USGS Core Research Center, 2018). Further digitization of this valuable resource resulting in core images will not only facilitate access to data for traditional analysis but also provide the information needed to build and calibrate innovative machine learning algorithms. The work we use here has the potential to organize many miles of slabbed cores into a reliable and coherent system easily accessible to a variety of users.

In this paper, we provide one of the first attempts to conduct automated core lithofacies classification using CNN. We begin with an overview of the methodology, which includes data preparation and transfer learning. The details of the CNN method are summarized in tutorial form in Appendix A. Next, we apply CNN to our core data set, and use confusion matrices, both test and validation accuracies, as well as a precision, recall, and F1 score (Fawcett, 2006) computed with the final-test set as means to analyze our results. We conclude with a summary of our findings and suggestions on how our workflow can be extended and improved.

Methodology

The deep learning methodology and CNN techniques are now very well disseminated in diverse fields. LeCun et al. (2015) presented details in the construction of and showed the value of deep learning. Dumoulin and Visin (2016) gave details on convolutions and other arithmetic steps used in deep learning algorithms. Although carefully constructed interative papers have been published detailing CNN image transformations and image understanding (e.g. Olah et al., 2017, 2018), CNN may appear to be "magic" and therefore somewhat suspect to the practicing geoscientist. For this reason, Appendix A provides a tutorial that looks under the covers, providing a simple CNN application to classify images into three groups. The work for this paper was developed using open-source computational packages described by Hunter (2007), Chollet et al. (2015), and Abadi et al. (2016)

When used for image recognition tasks, CNN models need examples (images) to understand the properties of each "class" they try to discriminate. Part of the parameters learned for a primary task (such as the ImageNet classification) can be transferred to a secondary task (e.g. lithofacies classification) through the use of transfer learning (Oquab et al., 2014; Pan and Yang, 2010; Yosinski et al., 2014). Our work focuses on using transfer learning of complex CNN architectures to serve our specific image recognition task. The following subsections detail how we prepared our datasets and give a brief explanation of transfer learning.

Data Preparation

We used core described using traditional methods published by Suriamin and Pranter (2018), capturing images using modern photographic equipment to generate the set of labeled data to feed our CNN. The total section used for this project consists of approximately 700 feet from one core from the Mississippian limestone and chert reservoirs in the Anadarko Shelf, Grant County, Oklahoma. The set of core images shown in Table 1 includes 17 different lithofacies. Two pairs of lithofacies exhibit similar lithology and appearance; we grouped these into a single class for this project (Figure 3). We carefully cropped the images in a standardized fashion, providing consistent input to the CNN. We used a sliding window technique to extract consistent squared cropped sections from the original core images (Figure 1), generating 180 by 180 pixel images representing roughly 2 by 2 inches of cored rock. Note in Figure 3 that the heavily damaged rock is not present in the images used for training/test. We chose to eliminate these images as they would increase variability within class. Ideally, more core data would provide sufficient images to define damaged classes. The sliding window cropping process augments the number of images of our initially small collection, thereby further generalizing the CNN. Some classes contained less than 300 images. In order to augment representation of those classes, we doubled the number of input images by flipping the image horizontally. Next, we select approximately 2% of the original data for each class to serve as test data. During training, 5% of the total training data is randomly selected to be part of training-test. The training-test set is used for an overall performance evaluation. We provide more detailed analysis using the test set. The selection of images to be part of the test has a higher standard than the images selected to be part of the training-test. As each image selected to be part of the test set force us to discard its neighbors (Figure 1), we select only 2% of the original data.

Even after image augmentation, Table 2 shows that some classes have a significantly larger number of images than others. This difference in amount of labeled data for different classes is referred as class imbalance (Buda et al., 2018; Japkowicz and Stephen, 2002) and can cause undesirable effects when training classifiers. In this study, we did not notice a significant bias caused by such class imbalance. Therefore, although we did not augment it, we chose to retain all of the images in the most common 13-14 Spiculitic mudstone-wackestone class. In contrast, we removed from analysis classes represented by less than 30 images where initial testing indicated that these under-sampled classes were reducing CNN accuracy.

Class	*Lithofacies*	*Training set*	*Test set*
01	Chert breccia in greenish shale matrix	*218	3
02	Chert breccia	*236	3
03	Skeletal mudstone-wackestone	*258	4
04	Skeletal grainstone	*160	3
05	Splotchy packstone grainstone	*344	4
06	Bedded skeletal peloidal packstone-grainstone	*416	4
07	Nodular packstone-grainstone	445	11
08	Skeletal peloidal packstone-grainstone	not used	not used
09	Bioturbated skeletal peloidal packstone-grainstone	795	19
10	Bioturbated mudstone-wackestone	*150	4
11	Brecciated spiculitic mudstone	not used	not used
12	Intraclast spiculitic mudstone	not used	not used
13 *14*	Spiculitic mudstone-wackestone Argillaceous spiculitic mudstone-wackestone	3077	79
15	Glauconitic sandstone	not used	not used
16 *17*	Shale Shaly claystone	789	17
Total number of images in each set		6888	151

Table 2: Class number assigned to each lithofacies in the core used in this study. Highlighted classes 13-14 and 16-17 exhibited similar lithology and appearance so combined to into two classes instead of four. During training, the training set data is further split: 10% are randomly selected to be part of a validation set and 5% are randomly assigned as training-test set. The proportion used for validation and test splitting are commonly dependent on the number of samples available and the type of machine learning model being trained. CNN models usually improve with more examples; therefore, we selected a smaller percentage to be part of the validation and test sets. Asterisks (*) indicate classes that were augmented using horizontally flipping the images. The last column of this table (Test set) comprises the selected images described in Figure 3 and is the test set used for further analysis in this paper. Classes with less than 30 original images were not used in this study (Modified from Suriamin and Pranter, 2018).

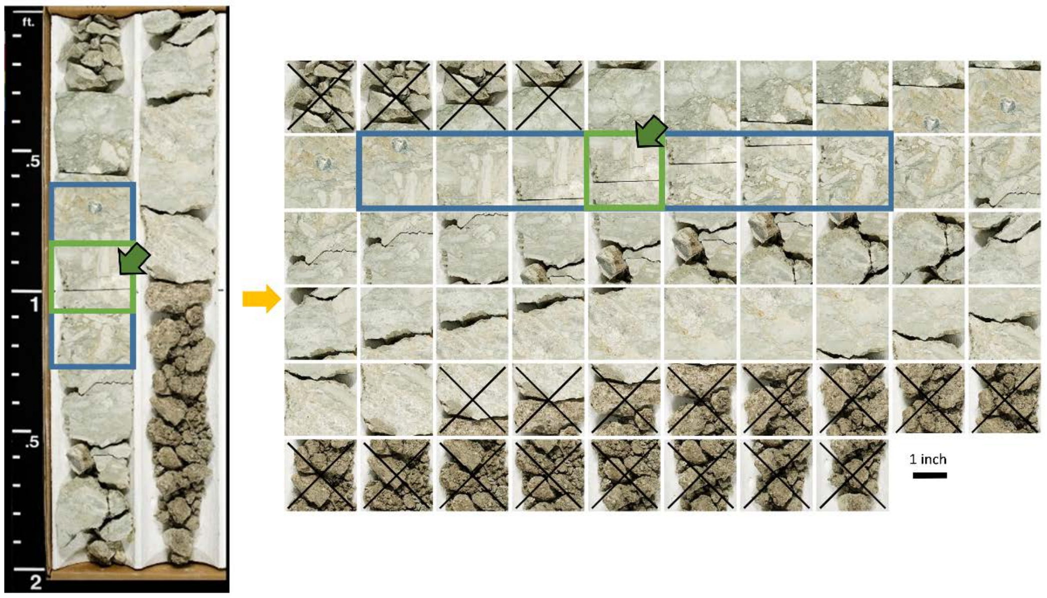


Figure 17: Figure showing image augmentation of a photographed core the core using a sliding window of cropped image. This approach provides the CNN with a greater amount of training data. The blue rectangle shows images that were never used during training (the test data). The cropped images crossed were discarded from the datasets (damaged rocks). The green arrow indicates a random image that could have been selected to be part of the test set. When an image like this is selected, the overlapping neighboring images are also removed from the training set. The separation of test data was the same for all classes in this project.

Transfer Learning

Transfer learning is a powerful technique that can be used to address the shortage of sufficient domain-specific training data (Carranza-Rojas et al., 2017). In transfer learning, the learned parameters of a base model trained on a base dataset are applied to a different task (Yosinski et al., 2014). In our application, we use a CNN model trained to identify the images of the ImageNet challenge to classify lithofacies in core (Figure 4). ImageNet is a dataset consisting of thousands of classes ranging from biological and household images to vehicles and bridges; to our knowledge no rock or core images were included in its construction. Another advantage of using transfer learning is to reduce the training computation time by using the trained layers as feature extractors (Appendix A) and rather training only a new classification layer. Examples of transfer learning include Carranza-Rojas et al. (2017) for herbarium specimens, Esteva et al. (2017) for skin cancer classification, and Gomez Villa et al. (2017) for camera-trap images. Tajbakhsh et al. (2016) used different medical imaging applications and performed a comparison between CNNs trained from scratch with the pre-trained CNNs. The authors found that the using a pre-trained CNN frequently outperforms a CNN model trained from scratch especially when limited training data are available.

When CNNs are trained with natural images, the first layers of the deep neural network learn features that are useful to identify textures or colors. This behavior is quite common in CNN models; the analysis is reevaluated if the initial layers learn image properties other than color or texture. Because of this CNN characteristic, models with good performance trained on the ImageNet challenge (e.g. Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2014, 2015; He et al., 2016; Zoph and Le, 2016; Zoph et al., 2017; Sandler et al., 2018) can be successfully retrained for new, field-specific classification problems (e.g. Tajbakhsh et al., 2016; Carranza-Rojas et al., 2017; Esteva et al., 2017; Gomez Villa et al., 2017; Norouzzadeh et al., 2018).

In this project, we evaluate transfer learning using four different trained models: InceptionV3 (Szegedy et al., 2015) consisting of 48 layers, ResNetV2 – implemented with 50 layers (He et al., 2016), MobileNetV2 (Sandler et al., 2018) with 20 layers, and NASNet (Zoph et al., 2017; Zoph and Le, 2016) with 20 layers. These models and the learned parameters are publicly available and can be downloaded from the TensorFlow Hub (2018) website. Each one of the CNN models require different sized images as input: 299 by 299 pixels for InceptionV3, 224 by 224 pixels for ResNetV2, 224 by 224 pixels for MobileNetV2, and 331 by 331 pixels for NASNet. Because our images are 180 by 180 pixels in size, we use simple bilinear interpolation to conform to the size of the transfer learning model used. As described in Appendix A, all subsequent layers are dependent on the size of the input data. Appendix A shows how transfer learning is achieved by simply using the convolutional layers as feature extractors for our core images thereby facilitating the training of a densely connected classification layer.

Carranza-Rojas et al. (2017), Esteva et al. (2017), and Gomez Villa et al. (2017) each used some 100,000 images in their datasets to perform transfer learning. Although we have a significantly smaller dataset consisting of less than7000 images, we still achieved a high-level accuracy as presented in the next section. We use confusion matrices, test and validation accuracy, and precision, recall, and F1 score (Fawcett, 2006) computed with the final-test set as means to analyze our results.

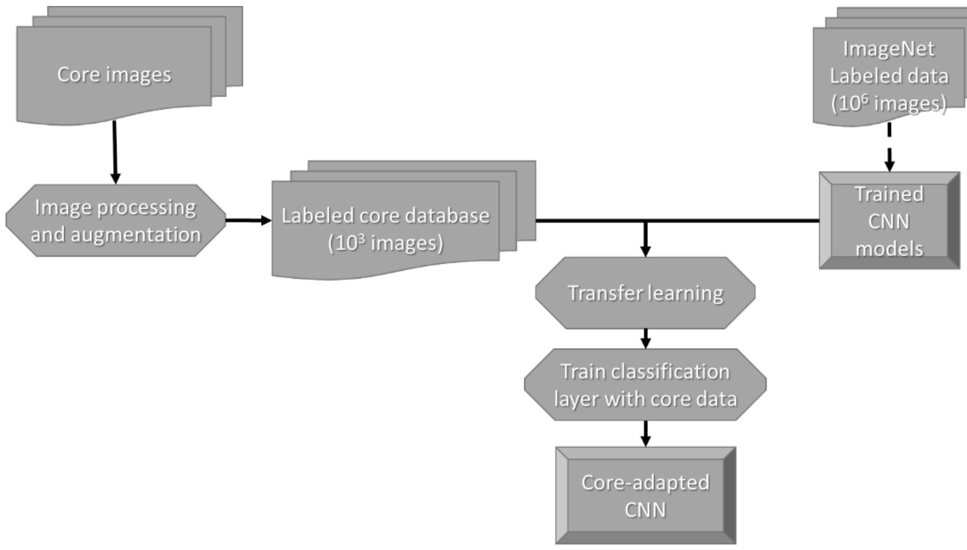

Core images
Image processing and augmentation
Labeled core database (10^3 images)
ImageNet Labeled data (10^6 images)
Trained CNN models
Transfer learning
Train classification layer with core data
Core-adapted CNN

Figure 18: Flowchart summarizing the workflow used in this paper. We begin with photographic images of the slabbed core, followed by simple image processing and data augmentation to generate our core image database. The CNN models we use as feature extractors were previously trained using millions of images on the ImageNet challenge. We then use transfer learning and re-use the ImageNet dataset classification CNN weights. Finally, we train the last layer to provide the desired core image classification.

Results

In this section we present the overall results we obtained as well as examples of the classifications performed by the retrained CNNs. Because the results of the four chosen CNNs are similar, we show in this section details of the ResNetV2 retrained CNN (apart from training-test set in Table 3). Plots and tables regarding the other three CNNs are presented in Appendix B.

The results for the training-test dataset are presented in Table 3.

Model	*Training-test Accuracy*
InceptionV3	0.95
ResNetV2	0.95
MobileNetV2	0.95
NASNet	0.90

Table 3: Training-test set data results for the different models used for transfer learning.

These training-tests accuracies were achieved after 5000 iterations using a gradient descent algorithm. Figure 3 shows the training and validation accuracy result for each step of the gradient descent. The CNN quickly reaches satisfactory levels of accuracy. After performing feature extraction, only the last classification layer needs to be trained. The training time in a single core CPU with 3.60 GHz maximum speed does not exceed one hour for our dataset for any of the four CNN models used. Access to graphical processing units (GPUs) provides even greater computation speeds.

Figure 6 shows representative images from the test dataset classified using the retrained ResNetV2. Although the CNN provides different levels of probability when assigning the classes allowing one to define an acceptable threshold to accept a given prediction, we simply assigned the image to the CNN class exhibiting the highest probability. Because the CNN provides different levels of probability when assigning the classes, we can define an acceptable threshold to accept a given prediction. Choosing different values for this threshold value is also a commonly used tool to analyze the performance of a classifying algorithm. Ferri et al. (2003), Everson and Fieldsend (2006), and Fawcett (2006) give details of the receiver operating characteristics graphs that arise when performing such analysis. In this paper, we choose the threshold to be 0.30; this means that we accept the image classification given by the CNN when any possible class receives a probability higher than 0.30. This value was chosen so that all images would be classified, even if the CNN is not very confident. Such threshold value is enough for our model to assign a class for each one of the images in the test set. Figure shows the confusion matrix generated when the test set is classified by the retrained ResNetV2. Precision, recall, F1 score, and support as well as weighted precision, recall, and F1 score are presented in Figure 5. All these metrics range from 0 (poor performance) to 1 (good performance). Precision and recall indicate how often the model was correct predicting the analyzed class. Precision is defined as ratio of true positives and the sum of true positives and false positives. Recall is defined as ratio between true positives and the sum of true positives and false negatives). F1 is the harmonic average of precision and recall.

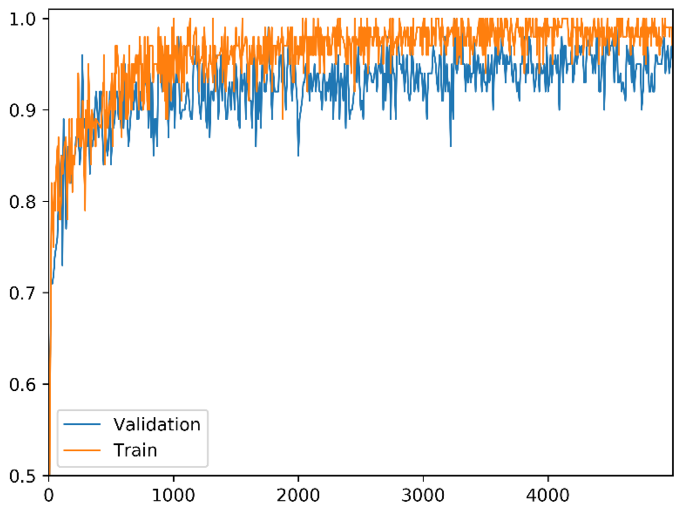


Figure 19: Validation and training accuracy for the ResNetV2 training. Note that after approximately 1000 iterations the gains are marginal. Because the cost of training the classification layer is inexpensive compared to training the entire model, we can afford to let the model train for many steps.

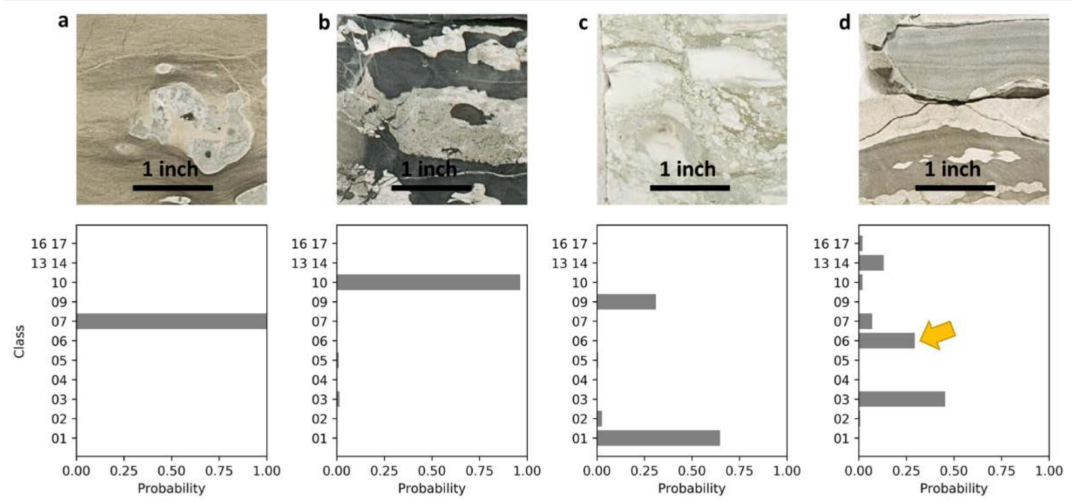


Figure 20: Examples of the classification performed by the retrained ResNetV2. (a)The CNN very confidently assigned the image to the correct class (class 07, Nodular packstone-grainstone). (b)Again the CNN provides a high level of confidence to assign the image to the correct class (class 10, Bioturbated mudstone-wackestone). (c) The CNN still assigns the image to the correct class, but with lower confidence (class 01, Chert breccia in greenish shale matrix is the correct class). (d) The image shows an example in which the CNN failed to correctly assign the class. The CNN assigned a higher confidence for class 03 (Skeletal mudstone-wackestone, with 0.45 probability) whereas the correct class is actually class 06 (Bedded skeletal peloidal packstone-grainstone, 0.29 probability, yellow arrow in the image). Setting a confidence threshold of 0.50 or greater would identify this classification as “ambiguous”, calling for human intervention.

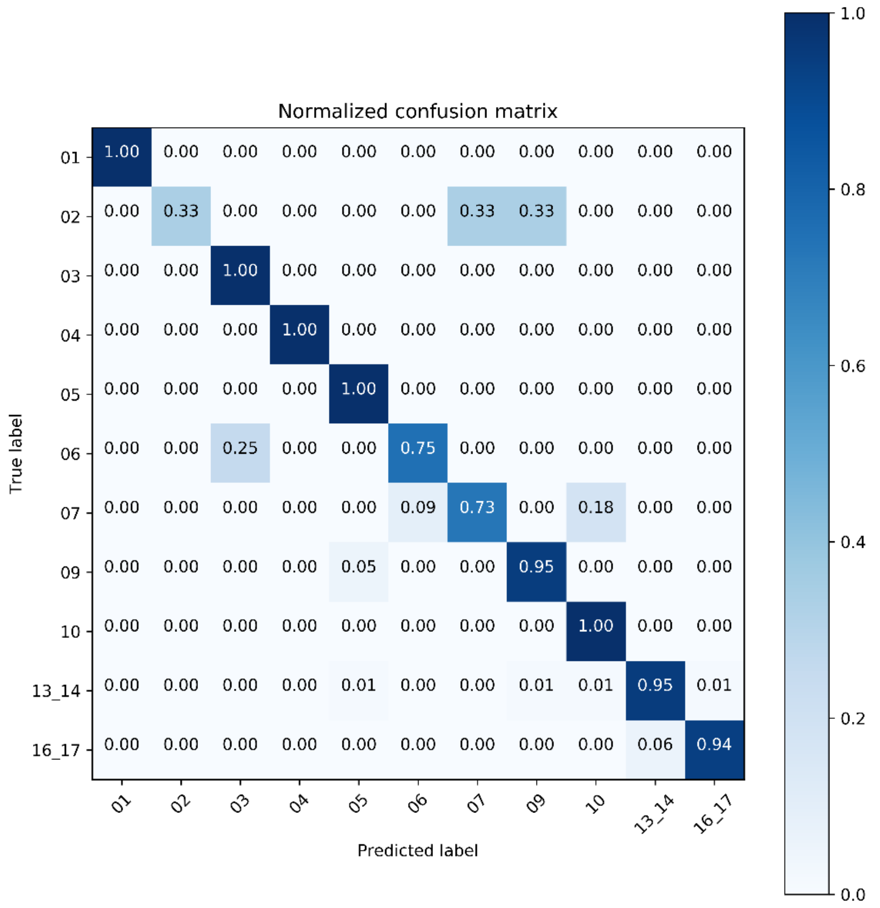


Figure 21: Normalized confusion matrix of the retrained ResNetV2 applied to the test set. Refer to Table 2 for class lithofacies and number of images for each class.

Class	*Precision*	*Recall*	*F1 score*	*Support*
01	1.00	1.00	1.00	3
02	1.00	0.33	0.50	3
03	0.80	1.00	0.89	4
04	1.00	1.00	1.00	3
05	0.67	1.00	0.80	4
06	0.75	0.75	0.75	4
07	0.89	0.73	0.80	11
09	0.90	0.95	0.92	19
10	0.57	1.00	0.73	4
13-14	0.99	0.95	0.97	79
16-17	0.94	0.94	0.94	17
weighted	0.93	0.92	0.92	

Table 4: Precision, recall, F1 score and support for the classification performed by the retrained ResNetV2. The last row shows the weighted values for each one of the metrics.

Discussion

To our knowledge this is the first study conducted using slabbed core image classification using CNNs. With comparable metrics for all the CNN architectures tested, we observe a high level of concordance (and confidence) between the expert labeled data and the classifications suggested by the CNNs. Using the methodology we presented in this paper, a user can obtain the probability that a standardized picture of a core belongs to one of the described lithofacies even if the user has little experience with core description. This capability can not only accelerate the interpretation of large data volumes by using non-expert technologists, but also identify inconsistencies in the interpretation between different experts working on the same data. This potential inconsistency suggests that we construct a human interpreter confusion matrix comparing multiple interpretations of the same core. This confusion matrix can then be constructed from a single interpreter and the CNN. Identification of such inconsistencies within teams composed of members with different backgrounds promises to facilitate data comprehension and accelerate project advancement. Even though we use a relatively small database of images, Figure 5 and Table 3 show the retrained ResNetV2 achieved high levels of accuracy. The remaining three architectures (InceptionV3, MobileNetV2, and NASNet) results also show high levels of accuracy (Appendix B).

When performing core description, a human interpreter relies on texture, structures, and pattern analysis to define the lithofacies being analyzed. In this manner, the classification performed by CNNs are somewhat mimicking human classification. Nonetheless, when a geologist is describing a rock, other rock properties (not visual properties) can be analyzed by the interpreter. Does the mineral react with acid? How hard is the mineral? Therefore, when using CNN models, the user needs to remember that the best result the CNN can provide is only the best result achieved by a visual (and strictly non-tactile) analysis of an image. We can, however, modify the deep learning architectures to be multidimensional. Specially, the digital images can be augmented by measures of resistivity, density, X-ray fluorescence, Fourier transform infrared spectroscopy, and other measures to produce an even more powerful tool.

In the architecture we used here, any image used as input to the CNN classifier will predict that the image belongs to one or more of the CNN's learned classes. This means that the CNN will never declare the image to be none of the pre-defined classes. *Figure 8* shows the picture of a carpet classified by the retrained ResNetV2. Although there are visual similarities between the carpet and the images in the training set, the resulting classification demonstrates the necessity of quality controlling CNNs output.

Because the CNN models are trained with expert labeled data, such expertise is abstractly maintained in the different parameters optimized in the CNN. Consequently, we can absorb interpretations performed by different specialists and save them in unique CNN models. Such data capture would provide a way of sharing geological knowledge great distances. Geoscientist working in challenging (or simply unfamiliar) geological settings can access the knowledge of a wide range of experts through the use of properly trained CNNs.

Human geoscientists will not be replaced by machine learning. Clearly, expert geoscientists are required to construct the labeled training data. Expert scientists are also required to quality control the prediction, perhaps manually examining all predictions that exhibit less than a threshold confidence. Emulating the approach of human geoscientists who subliminally apply models of deposition and diagenesis while examining core, will be very difficult. Linking different lithologies within a parasequence is part of human interpretation. At present, computers have a difficult time with such geoscientific image segmentation problems.

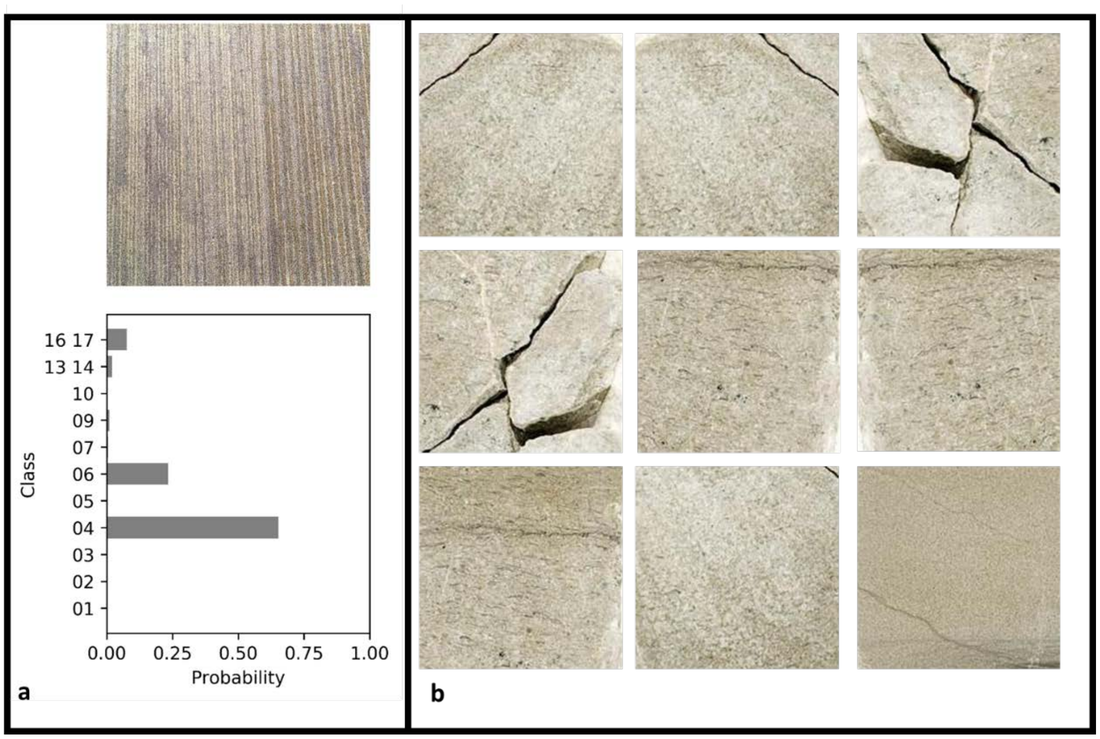


Figure 22: (a) A photographic image of a carpet classified by the ResNetV2, and (b) examples of images from the class 4 training dataset. The CNN is 70% confident that the carpet belongs to class 4 - Skeletal grainstone.

Suggestions for further study

During the course of the work we had access to pictures of a single core and we presented the results we obtained using modern CNN models for classification for that single interpreted core. We envision the process used in this paper can be used to greatly accelerate the interpretation of multiple cores. Users can achieve such multi-core interpretation result with a more iterative approach: an experienced expert labels the key lithofacies of the region; the CNN is then trained and classifies the remaining cores. The results of such classification are then evaluated by the expert - a form of active learning (Sener and Savarese, 2018; Settles, 2012). If necessary, the user can retrain the CNN with a now increased set of labeled images (the originally expert-labeled images and the new CNN-labeled images). In this manner, many miles of core can be interpreted with lower effort. Several new challenges can arise when working with historical data of lesser quality, different formats, with different interpretations, and from different well locations sampling different geology. This paper shows one successful application of a growing technology, however different evaluations need to be addressed for every specific task. When these extra variables, such as poor-quality data or multiple wells with inconsistent interpretations, it is likely that the performance will be negatively affected.

For this project we relied on standardized core pictures and a simple sliding window to extract images. Therefore, it is reasonable to assume that some images will show more than one lithofacies. Different ways of data acquisition can further improve the results of CNN models.

Conclusions

In this paper, we provide one of the first attempts to conduct automated core lithofacies classification using CNNs. The methodology we use does not depend on specialized bench work and can be applied to existing images of slabbed cores.

Efforts in data digitization are important initiatives to preserve scientific knowledge and the approach we use here can be improved with information generated from such endeavors. The development of customized core-databases can be of extreme value for companies and researchers that have work dependent on core description. When operators need to reevaluate prospective plays -due to new acreage acquisition or to update the geological knowledge with modern geological information- thousands of feet of expensive slabbed core might be overlooked due to time and personnel constraints. Further development of the project we present here will ultimately speed up the process of core description with the use of slabbed core specific CNN models.

Using the technology we applied in this paper, an experienced geologist describes a small percentage of the core using traditional, careful, and standardized “visual and tactile” lithologic description and then uses such information to train a CNN. The trained CNN can then classify the remaining core -the geologist can quality control the results and will have more time to work on the necessary details. The methodology we used here can be used to standardize interpretation in large collections of core data. As interpretation might be subjective, teams can choose to maintain “the best” interpreter knowledge abstractly captured by a CNN trained with data labeled by the most experienced geologist, or to train a CNN using only new concepts. A task that would be infeasible now (having different specialists interpreting miles of core) can be achieved with the help of CNN models.

Acknowledgments

Rafael acknowledges CNPq (grant 203589/2014-9) for graduate sponsorship and CPRM for granting absence of leave. We thank TensorFlow (https://www.tensorflow.org) and Keras communities (https://keras.io) for making valuable deep learning tools available to the public. The funding for this project was provided by the industry sponsors of the OU Attribute-Assisted Seismic Processing and Interpretation (AASPI) Consortium. The funding for the original core analysis was provided by the industrial sponsors of the OU Mississippi Lime Consortium. We thank the anonymous reviewers that helped us improve the quality of this manuscript. We are thankful for Irene Espejo's support and careful review of the manuscript.

Appendix A

Convolutional neural networks intuitions

Although deep learning and convolutional neural networks (CNN) seem to have become buzzwords, the intuition of how these techniques work might be obscured for some of us. Although CNN models are becoming increasingly complexes, the building blocks are very familiar to us geoscientists. Convolutions – whereas performed in one, two, three, or n-dimensions - are the same operations we become familiarized when dealing with a seismic wavelet convolving with the Earth's reflectivity series. Many seismic attributes are also based on convolutions. Innumerous apps and software offer the user the option to extract the edges of an image – or to blur such image. These are just a few of many convolution operations we commonly encounter. Our objective with this appendix is to give a short and informal overview of the essentials of CNNs.

When using CNNs for image classification tasks, the models use the resulted filtered image (an image convolved with a convolution kernel) as input to another operation (or the next layer). The deep learning nomenclature comes from this pattern – the input of a layer is used as input to the next one.

In this appendix, we have an easy task for a CNN: to classify three classes of very distinct images (Figure A- *1*). Each class has 10 examples of RGB images 180 x 180 pixels that were extracted from pictures of a slabbed core in the Mississippian limestone and chert reservoirs. Geologists can easily tell the difference between the classes and could probably correctly name the lithofacies even with these low-resolution images. The CNN actually needs the interpreter (the domain expert) to correctly label the images and separate them – in our simple example we are simply calling the lithofacies class 1, class 2, and class 3.

We design a not-so-deep CNN showed in Figure A- *2*. This CNN is composed of six layers: convolution, max pooling, convolution, max pooling, flatten, and dense layers. The pooling layer extracts statistics of submatrices of the input data. In this case, we are using maximum (max), therefore these operations look at a submatrix of the input and keeps the maximum value of that submatrix to be the input for the next layer. The flatten layer simply restructures the data to be a single column vector. The dense layer is the traditional neural network composed a linear transformation followed by a nonlinear transformation (softmax in this case) densely connected, i.e. each one of the elements is connected to each one of the neurons in the upcoming neuron. The output of the dense layer is the probability an image belongs to classes 1, 2, or 3.

During training all the randomly started parameters are optimized to reduce the cost function. The cost function is commonly defined as the sum of the loss/error of an image being assigned in the wrong class in the training set. This simple example has only a training set, we did not set validation and test set as would be appropriate for a real machine learning methodology. Therefore, after training we have this set of parameters that can take an image, perform different operations on such image and come up with a value of how probable the image belongs to one of the training classes. This image transformation is displayed in *Figure A- 3*. Note the CNN very confidently sets the image as belonging to class 3 (with 1.00 confidence). This result can be achieved because the classes are very well defined, and the images have low variance (all images in the same class are very similar to each other). As can be seen in *Figure A-*

3, this small CNN trained convolutional kernels that are very good to detect edges. If we wanted to use transfer learning with this CNN, we would "delete" the last layer (softmax, gold hexagon d) and add a new classification layer.

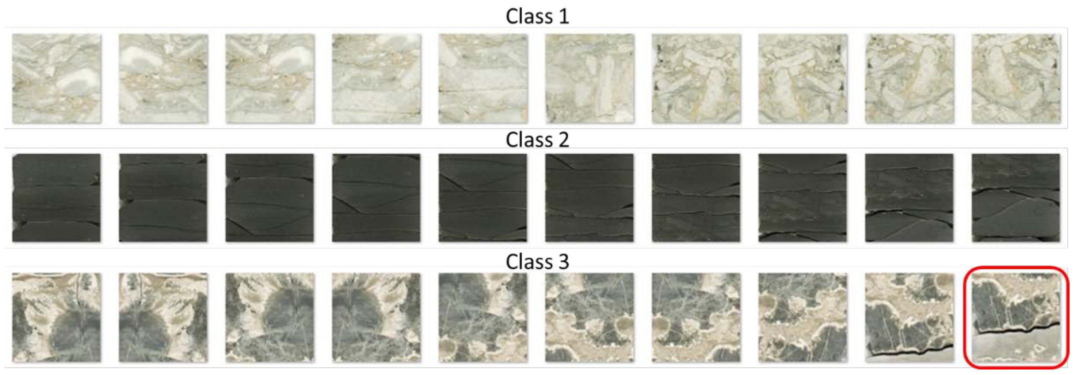


Figure A- 1: Our very simple set of images used in toy CNN example. We highlighted the image in class 3 that is used as example in Figure A- 3. As all images are very similar in their set, this is an easy task for CNN models and we can achieve high accuracy with a simple network.

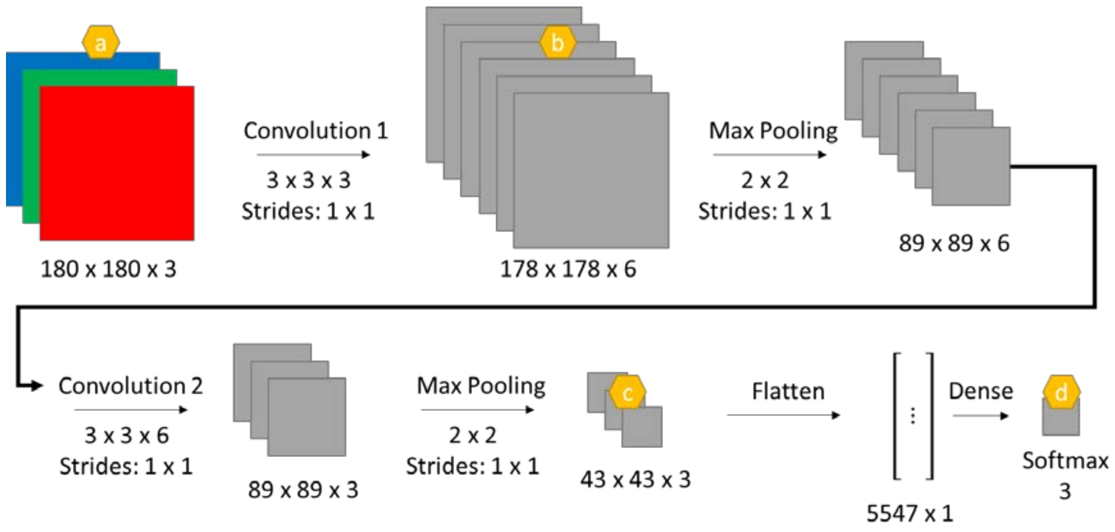


Figure A- 2: A simple convolutional neural network. The golden hexagons show images displayed in figure next. In this toy example, a set of images with size 180 x 180 pixels is input to a CNN with six layers. The first layer is a set of six convolution kernels with size 3 x 3 x 3. The value of the third dimension is the same as the value of the number of channels of the previous layer. Note that after the first convolution the object reduces in height and width, but its number of channels increases. For the next step, a max pooling (an operation in which we extract the maximum value of a submatrix of the input) further reduces the height and width. This “thinner” object is then input to another convolution layer following by another max pooling. After the last max pooling, the layer is then flattened, meaning all its values are stored as a single vector. The last layer uses as input all the values of the flattened vector to compute the probability that the input image belongs to one of three classes. Note that with this architecture whatever is used as input will output some probability of belonging to one of the three classes. The kernels of the convolutional layers and the softmax of the last layer are the parameters that need to be trained for this neural network. In this example, we need to train a total of 16977 parameters. For Convolution 1, we need to train 3 x 3 x 3 x 6 + 6 (bias) = 168 parameters. For Convolution 2, 165 parameters. The dense layer is responsible for 16644 parameters that need training. These ratios (convolution and dense parameters) should not be used as comparison with more complexes CNNs as the one we used in the main body of this study, truly deep CNN will have many more convolution parameters to be optimized.

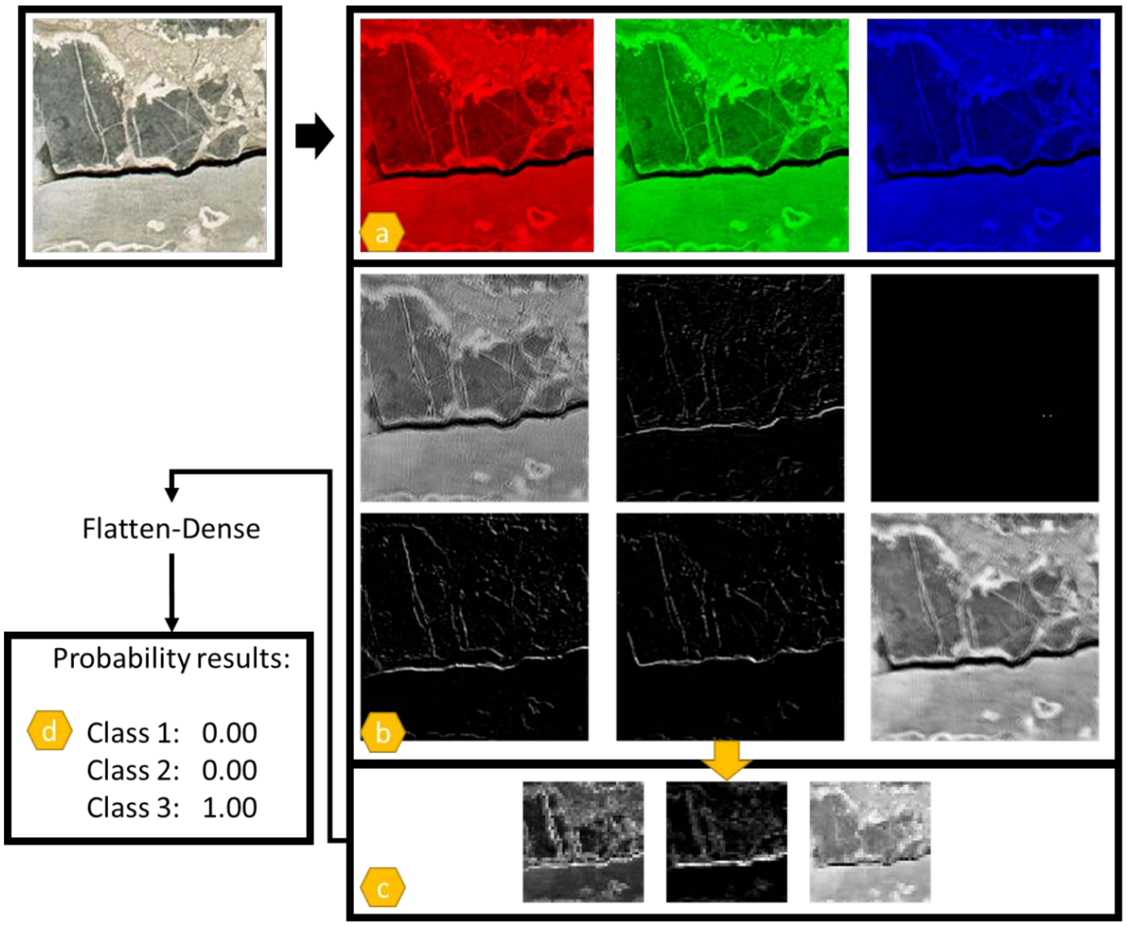


Figure A- 3: Simplified workflow and resulting images extracted from different layers when the figure on top left is input to the CNN showed in Figure A- 2 after training. The golden hexagons can be used for easier reference between this figure and Figure A- 2. Note how the set of weights (the convolutional kernels) learned by this CNN learns how to identify edges in the input image. This is a common behavior in CNN when used with natural images (Yosinski et al., 2014). In a sense, much as a trained geologist, the CNN learns how to identify different patterns. Note the image in frame a is a simple decomposition of the original image, therefore we choose to display them as red-green-blue color. The images in frames b and c are results of different “filters” applied in different steps of the CNN and are composed of single channel (the convolution kernels have different sizes as showed in Figure A- 2) . We choose to display these images as grayscale.

Appendix B

Inceptionv3, mobilenetv2, and nasnet metrics

In this appendix we show the metrics for the retrained CNN models not presented in the main text.

InceptionV3

Train and validation accuracy result for each step of the gradient descent are presented in Figure B- *1*. Figure B- *2* shows the confusion matrix generated when the test set is classified by the retrained InceptionV3. Precision, recall, F1 score, and support as well as weighted precision, recall, and F1 score are presented in Table B- *1*.

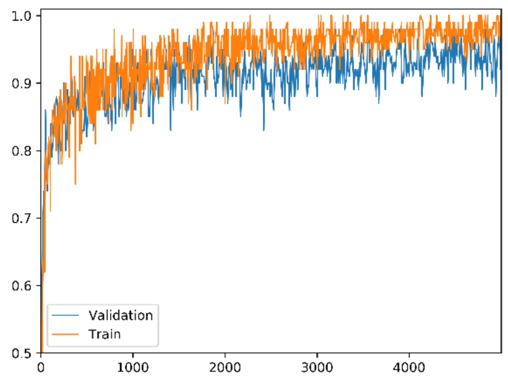
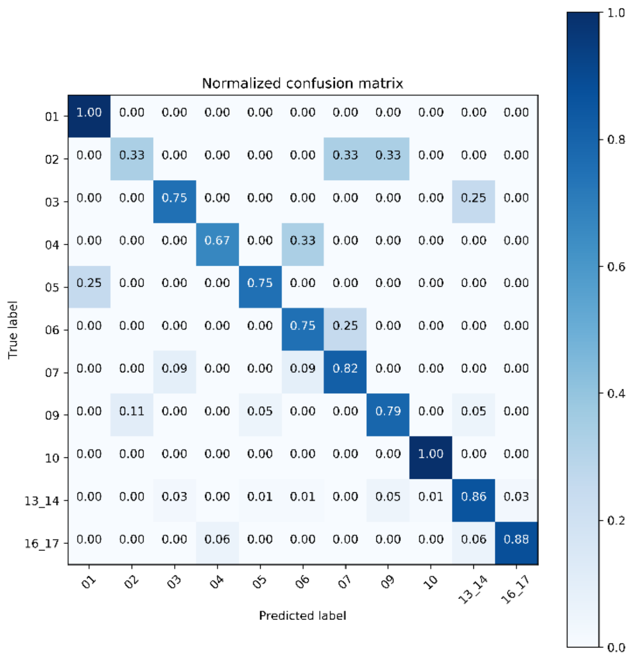


Figure B- 1: Validation and train accuracy for the InceptionV3 training. Note that after approximately 2000 iterations the gains are marginal. As the cost of training the classification layer is inexpensive, we can afford to let the model train for many steps.



	01	02	03	04	05	06	07	09	10	13_14	16_17
01	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
02	0.00	0.33	0.00	0.00	0.00	0.00	0.33	0.33	0.00	0.00	0.00
03	0.00	0.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.00
04	0.00	0.00	0.00	0.67	0.00	0.33	0.00	0.00	0.00	0.00	0.00
05	0.25	0.00	0.00	0.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00
06	0.00	0.00	0.00	0.00	0.00	0.75	0.25	0.00	0.00	0.00	0.00
07	0.00	0.00	0.09	0.00	0.00	0.09	0.82	0.00	0.00	0.00	0.00
09	0.00	0.11	0.00	0.00	0.05	0.00	0.00	0.79	0.00	0.05	0.00
10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
13_14	0.00	0.00	0.03	0.00	0.01	0.01	0.00	0.05	0.01	0.86	0.03
16_17	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.06	0.88

Figure B- 2: Normalized confusion matrix of the retrained InceptionV3 applied to the test set. Refer to Table 2in main text for class lithofacies and number of images for each class.

Table B- 1: Precision, recall, F1 score and support for the classification performed by the retrained InceptionV3. Last row shows the weighted values for each one of the metrics.

Class	*Precision*	*Recall*	*F1 score*	*Support*
01	0.75	1.00	0.86	3
02	0.33	0.33	0.33	3
03	0.50	0.75	0.60	4
04	0.67	0.67	0.67	3
05	0.60	0.75	0.67	4
06	0.50	0.75	0.60	4
07	0.82	0.82	0.82	11
09	0.75	0.79	0.77	19
10	0.80	1.00	0.89	4
13-14	0.96	0.86	0.91	79
16-17	0.88	0.88	0.88	17
weighted	0.85	0.83	0.84	

MobileNetV2

Train and validation accuracy result for each step of the gradient descent are presented in Figure B- *3*. Figure B- *4* shows the confusion matrix generated when the test set is classified by the retrained MobileNetV2. Precision, recall, F1 score, and support as well as weighted precision, recall, and F1 score are presented in Table B- *2*.

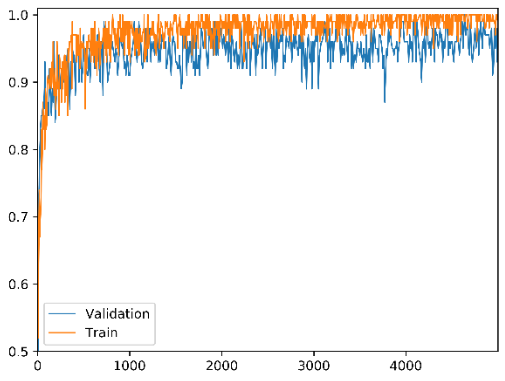


Figure B- 3: Validation and train accuracy for the MobilNetV3 training. Note that after approximately 1000 iterations the gains are marginal. As the cost of training the classification layer is inexpensive, we can afford to let the model train for many steps.

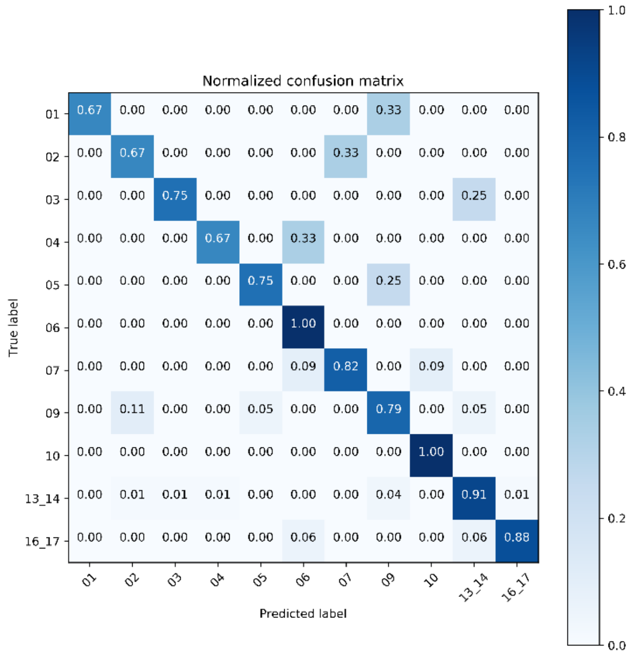


Figure B- 4: Normalized confusion matrix of the retrained MobileNetV2 applied to the test set. Refer to Table 2 in main text for class lithofacies and number of images for each class.

Table B- 2: Precision, recall, F1 score and support for the classification performed by the retrained MobileNetV2. Last row shows the weighted values for each one of the metrics.

Class	*Precision*	*Recall*	*F1 score*	*Support*
01	1.00	0.67	0.80	3
02	0.40	0.67	0.50	3
03	0.75	0.75	0.75	4
04	0.67	0.67	0.67	3
05	0.75	0.75	0.75	4
06	0.57	1.00	0.73	4
07	0.90	0.82	0.86	11
09	0.75	0.79	0.77	19
10	0.80	1.00	0.89	4
13-14	0.96	0.91	0.94	79
16-17	0.94	0.88	0.91	17
weighted	0.89	0.87	0.87	

NASNet

Train and validation accuracy result for each step of the gradient descent are presented in Figure B- *1*. *Figure B- 6* shows the confusion matrix generated when the test set is classified by the retrained NASNet. Precision, recall, F1 score, and support as well as weighted precision, recall, and F1 score are presented in *Table B- 3*.

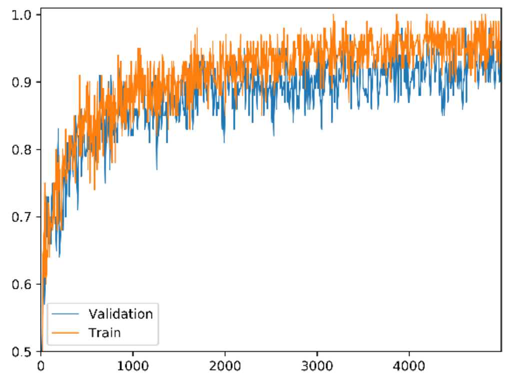


Figure B- 5: Validation and train accuracy for the NASNet training. Note that this architecture takes longer to increase its accuracy.

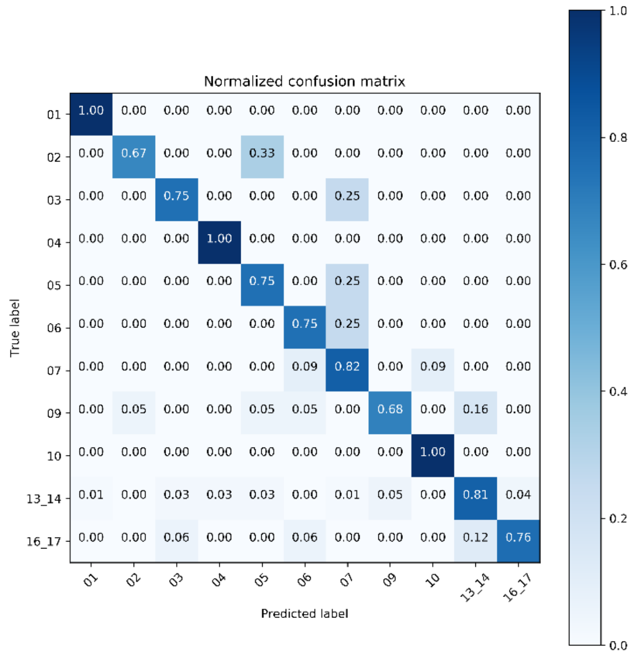


Figure B- 6: Normalized confusion matrix of the retrained NASNet applied to the test set. Refer to Table 2 in main text for class lithofacies and number of images for each class.

Table B- 3: Precision, recall, F1 score and support for the classification performed by the retrained NASNet. Last row shows the weighted values for each one of the metrics.

Class	*Precision*	*Recall*	*F1 score*	*Support*
01	0.75	1.00	0.86	3
02	0.67	0.67	0.67	3
03	0.50	0.75	0.60	4
04	0.60	1.00	0.75	3
05	0.43	0.75	0.55	4
06	0.50	0.75	0.60	4
07	0.69	0.82	0.75	11
09	0.77	0.68	0.72	19
10	0.80	1.00	0.89	4
13-14	0.93	0.81	0.87	79
16-17	0.81	0.77	0.79	17
weighted	0.82	0.80	0.80	

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X., 2016. TensorFlow: A system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). pp. 265–283.

Allen, D.B., Pranter, M.J., 2016. Geologically constrained electrofacies classification of fluvial deposits: An example from the Cretaceous Mesaverde Group, Uinta and Piceance Basins. Am. Assoc. Pet. Geol. Bull. 100, 1775–1801. https://doi.org/10.1306/05131614229

Baldwin, J.L., Bateman, R.M., Wheatley, C.L., 1990. Application of a neural network to the problem of mineral identification from well logs, The Log Analyst. Society of Professional Well Log Analysts.

Bestagini, P., Lipari, V., Tubaro, S., 2017. A machine learning approach to facies classification using well logs, in: SEG Technical Program Expanded Abstracts 2017. Society of Exploration Geophysicists, pp. 2137–2142. https://doi.org/10.1190/segam2017-17729805.1

Blagoderov, V., Kitching, I.J., Livermore, L., Simonsen, T.J., Smith, V.S., 2012. No specimen left behind: industrial scale digitization of natural history collections. Zookeys 133–46. https://doi.org/10.3897/zookeys.209.3178

Buda, M., Maki, A., Mazurowski, M.A., 2018. A systematic study of the class imbalance problem in convolutional neural networks. Neural Networks 106, 249–259. https://doi.org/10.1016/J.NEUNET.2018.07.011

Carranza-Rojas, J., Goeau, H., Bonnet, P., Mata-Montero, E., Joly, A., 2017. Going deeper in the automated identification of Herbarium specimens. BMC Evol. Biol. 17, 181. https://doi.org/10.1186/s12862-017-1014-z

Chollet, F., others, 2015. Keras.

Daily, M., Medasani, S., Behringer, R., Trivedi, M., 2017. Self-Driving Cars. Computer (Long. Beach. Calif). 50, 18–23. https://doi.org/10.1109/MC.2017.4451204

de Matos, M.C., Osorio, P.L., Johann, P.R., 2007. Unsupervised seismic facies analysis using wavelet transform and self-organizing maps. Geophysics 72, P9–P21. https://doi.org/10.1190/1.2392789

de Matos, M.C., Yenugu, M. (Moe), Angelo, S.M., Marfurt, K.J., 2011. Integrated seismic texture segmentation and cluster analysis applied to channel delineation and chert reservoir characterization. Geophysics 76, P11–P21. https://doi.org/10.1190/geo2010-0150.1

Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L., 2009. ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Dumoulin, V., Visin, F., 2016. A guide to convolution arithmetic for deep learning. ArXiv e-prints.

Ellwood, E.R., Dunckel, B.A., Flemons, P., Guralnick, R., Nelson, G., Newman, G., Newman, S., Paul, D., Riccardi, G., Rios, N., Seltmann, K.C., Mast, A.R., 2015. Accelerating the Digitization of Biodiversity Research Specimens through Online Public Participation. Bioscience 65, 383–396. https://doi.org/10.1093/biosci/biv005

Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S., 2017. Dermatologist-level classification of skin cancer with deep neural networks. Nature 542, 115–118. https://doi.org/10.1038/nature21056

Everson, R.M., Fieldsend, J.E., 2006. Multi-class ROC analysis from a multi-objective optimisation perspective. Pattern Recognit. Lett. 27, 918–927. https://doi.org/10.1016/J.PATREC.2005.10.016

Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognit. Lett. 27, 861–874. https://doi.org/10.1016/J.PATREC.2005.10.010

Ferri, C., Hernández-Orallo, J., Salido, M.A., 2003. Volume under the ROC Surface for Multi-class Problems BT - Machine Learning: ECML 2003, in: Lavrač, N., Gamberger, D., Blockeel, H., Todorovski, L. (Eds.), . Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 108–120.

Gardner, J.W., Pearce, T.C., Friel, S., Bartlett, P.N., Blair, N., 1994. A multisensor system for beer flavour monitoring using an array of conducting polymers and predictive classifiers. Sensors Actuators B Chem. 18, 240–243. https://doi.org/10.1016/0925-4005(94)87089-6

Gomez Villa, A., Salazar, A., Vargas, F., 2017. Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. Ecol. Inform. 41, 24–32. https://doi.org/10.1016/J.ECOINF.2017.07.004

He, K., Zhang, X., Ren, S., Sun, J., 2016. Identity Mappings in Deep Residual Networks, in: Leibe, B., Matas, J., Sebe, N., Welling, M. (Eds.), Computer Vision -- ECCV 2016. Springer International Publishing, Cham, pp. 630–645.

Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Comput. Sci. Eng. 9, 90–95. https://doi.org/10.1109/MCSE.2007.55

Japkowicz, N., Stephen, S., 2002. The Class Imbalance Problem: A Systematic Study. Intell. Data Anal. 6, 429–449.

Kortström, J., Uski, M., Tiira, T., 2016. Automatic classification of seismic events within a regional seismograph network. Comput. Geosci. 87, 22–30. https://doi.org/10.1016/J.CAGEO.2015.11.006

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet Classification with Deep Convolutional Neural Networks, in: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12. Curran Associates Inc., USA, pp. 1097–1105.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444. https://doi.org/10.1038/nature14539

Norouzzadeh, M.S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M.S., Packer, C., Clune, J., 2018. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. Proc. Natl. Acad. Sci. U. S. A. 115, E5716–E5725. https://doi.org/10.1073/pnas.1719367115

Olah, C., Mordvintsev, A., Schubert, L., 2017. Feature Visualization. Distill. https://doi.org/10.23915/distill.00007

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., Mordvintsev, A., 2018. The Building Blocks of Interpretability. Distill. https://doi.org/10.23915/distill.00010

Oquab, M., Bottou, L., Laptev, I., Sivic, J., 2014. Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, pp. 1717–1724. https://doi.org/10.1109/CVPR.2014.222

Pan, S.J., Yang, Q., 2010. A Survey on Transfer Learning. IEEE Trans. Knowl. Data Eng. 22, 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Perol, T., Gharbi, M., Denolle, M., 2018. Convolutional neural network for earthquake detection and location. Sci. Adv. 4, e1700578. https://doi.org/10.1126/sciadv.1700578

Qi, J., Lin, T., Zhao, T., Li, F., Marfurt, K., 2016. Semisupervised multiattribute seismic facies analysis. Interpretation 4, SB91–SB106. https://doi.org/10.1190/INT-2015-0098.1

Qian, F., Yin, M., Liu, X.-Y., Wang, Y.-J., Lu, C., Hu, G.-M., 2018. Unsupervised seismic facies analysis via deep convolutional autoencoders. Geophysics 83, A39–A43. https://doi.org/10.1190/geo2017-0524.1

Roy, A., Romero-Peláez, A.S., Kwiatkowski, T.J., Marfurt, K.J., 2014. Generative topographic mapping for seismic facies estimation of a carbonate wash, Veracruz Basin, southern Mexico. Interpretation 2, SA31–SA47. https://doi.org/10.1190/INT-2013-0077.1

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vis. 115, 211–252. https://doi.org/10.1007/s11263-015-0816-y

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. ArXiv e-prints.

Sener, O., Savarese, S., 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach, in: International Conference on Learning Representations.

Settles, B., 2012. Active Learning. Synth. Lect. Artif. Intell. Mach. Learn. 6, 1–114. https://doi.org/10.2200/S00429ED1V01Y201207AIM018

Simonyan, K., Zisserman, A., 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv e–prints.

Sinha, S., Wen, Y., Pires de Lima, R.A., Marfurt, K., 2018. Statistical controls on induced seismicity. Unconventional Resources Technology Conference. https://doi.org/10.15530/urtec-2018-2897507-MS

Suriamin, F., Pranter, M.J., 2018. Stratigraphic and lithofacies control on pore characteristics of Mississippian limestone and chert reservoirs of north-central Oklahoma. Interpretation 1–66. https://doi.org/10.1190/int-2017-0204.1

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2014. Going Deeper with Convolutions. CoRR abs/1409.4.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2015. Rethinking the Inception Architecture for Computer Vision. CoRR abs/1512.0.

Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B., Liang, J., 2016. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? IEEE Trans. Med. Imaging 35, 1299–1312. https://doi.org/10.1109/TMI.2016.2535302

TensorFlow Hub [WWW Document], 2018. URL https://tfhub.dev/ (accessed 11.26.18).

Wu, Y., Lin, Y., Zhou, Z., Bolton, D.C., Liu, J., Johnson, P., 2018. DeepDetect: A Cascaded Region-Based Densely Connected Network for Seismic Event Detection. IEEE Trans. Geosci. Remote Sens. 1–14. https://doi.org/10.1109/TGRS.2018.2852302

Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks? Adv. Neural Inf. Process. Syst. 27, 3320–3328.

Zhang, G., Wang, Z., Li, H., Sun, Y., Zhang, Q., Chen, W., 2018. Permeability prediction of isolated channel sands using machine learning. J. Appl. Geophys. 159, 605–615. https://doi.org/10.1016/J.JAPPGEO.2018.09.011

Zhang, Y., Salisch, H.A., McPherson, J.G., 1999. Application of neural networks to identify lithofacies from well logs. Explor. Geophys. 30, 45. https://doi.org/10.1071/EG999045

Zhao, T., Li, F., Marfurt, K.J., 2017. Constraining self-organizing map facies analysis with stratigraphy: An approach to increase the credibility in automatic seismic facies classification. Interpretation 5, T163–T171. https://doi.org/10.1190/INT-2016-0132.1

Zhao, T., Zhang, J., Li, F., Marfurt, K.J., 2016. Characterizing a turbidite system in Canterbury Basin, New Zealand, using seismic attributes and distance-preserving self-organizing maps. Interpretation 4, SB79–SB89. https://doi.org/10.1190/INT-2015-0094.1

Zoph, B., Le, Q. V, 2016. Neural Architecture Search with Reinforcement Learning. CoRR abs/1611.0.

Zoph, B., Vasudevan, V., Shlens, J., Le, Q. V, 2017. Learning Transferable Architectures for Scalable Image Recognition. CoRR abs/1707.0.

Chapter 4: Convolutional neural networks as an aid to biostratigraphy: A test on Late Paleozoic microfossils

Rafael Pires de Lima[1,2], Katie F. Welch[1], James E. Barrick[3], Kurt J. Marfurt[1], Gerilyn S. Soreghan[1], Roger Burkhalter[4], Murphy Cassel[5]

[1]School of Geology and Geophysics, The University of Oklahoma, 100 East Boyd Street, RM 710, Norman, Oklahoma, 73019, USA

[2]The Geological Survey of Brazil – CPRM, 55 Rua Costa, São Paulo, São Paulo, Brazil

[3]Department of Geosciences, Mail Stop 1053, Texas Tech University, Lubbock, Texas, 79409, USA

[4]Sam Nobel Museum, The University of Oklahoma, 2401 Chautauqua Ave., Norman, Oklahoma, 73072, USA

[5]Halliburton, 14206 Maranta Estates court, Cypress, Texas, 77429, USA

Preface

This chapter was reworked and is presented here as and it was when resubmitted for publication in a sedimentary geology journal. This study was originated by a questioning from Dr. Lynn Soreghan during Murphy Cassel’s Master’s Thesis defense about fossil attributes. High quality thin-sections of Fusulinid specimens are hard to obtain and can provide invaluable information that can be used for dating through biostratigraphy as well as paleoenvironmental conditions, yet access to experts to experts can lead to an overlook of such microfossils. The results presented in this chapter show that convolutional neural networks can be used to classify Fusulinid thin-sections.

Abstract

Accurate taxonomic classification of microfossils in thin-sections is an important biostratigraphic procedure. As paleontological expertise is often restricted to specific taxonomic groups and experts are not present in all institutions, geoscience researchers often suffer from lack of quick access to critical taxonomic knowledge for biostratigraphic analyses. Moreover, diminishing emphasis on education and training in systematics poses a major challenge for the future of biostratigraphy and on associated endeavors reliant on systematics. Here we present a machine learning approach to classify and organize fusulinids—microscopic index fossils for the Late Paleozoic. The technique we employ has the potential to use such important taxonomic knowledge in models that can be applied to recognize and categorize fossil specimens. Our results demonstrate that, given adequate images and training, convolutional neural network models can correctly identify fusulinids with high levels of accuracy. Continued efforts in digitization of biological and paleontological collections at numerous museums and adoption of machine learning by paleontologists can enable the development of highly accurate and easy-to-use classification tools and, thus, facilitate biostratigraphic analyses by non-experts as well as allow for cross-validation of disparate collections around the world.

Plain Language Summary

This research focuses on the applications of convolutional neural networks (CNN) as a means to facilitate fossil classification. CNN is an artificial neural network architecture commonly used for image classification tasks. By using test-case thin-sections of fusulinids we found that CNN can categorize images of fossilized specimens with a high level of accuracy. We present one of the few applications of CNN using bisected fossil specimens identified specifically through 2D thin-sections to perform genus and species identification.

Introduction

Biostratigraphy is a critical approach for dating and correlating sedimentary successions, particularly given the common absence of material appropriate for radioisotopic dating of sedimentary strata. Biostratigraphy hinges on detailed analysis of extracted fossils, or thin-sections of fossils, to identify specimens to the genus and species levels. In addition to their utility for dating, fossil assemblages shed light on paleoenvironmental conditions; foraminiferal assemblages, for example, can yield information critical for reconstructing histories of paleoclimatic and palaeoceanographic conditions (e.g., Gooday, 1994; Culver, 2003; Kucera, 2007; Roozpeykar and Moghaddam, 2016). Commonly, biostratigraphy relies on species-level identification of well-established microfossil groups such as foraminifera, conodonts, and palynomorphs. The complex morphology of fossil organisms has required the use of specialists for reliable and correct systematics, especially for the generation of detailed and accurate biostratigraphic correlation. Unfortunately, education and training in the identification of fossil taxa is diminishing, greatly crippling future capacity in this area (Farley and Armentrout, 2000, 2002).

Although the largest investment of resources, both time and financial, for biostratigraphic studies may be for data acquisition and sample preparation, not every institution has the necessary expert to make accurate species-level identification for biostratigraphic analyses. Both old and new paleontological collections of significant biostratigraphic value may be overlooked or ignored because no one is available to perform the necessary taxonomic identifications. Because of this paucity of experts, a procedure is needed that can help facilitate the access of existing taxonomic knowledge to a broader audience. If a tool with reliable accuracy can be used to identify different taxonomic groups, a wider range of geoscientists can rely on years of accumulated, but difficult-to-access, paleontological knowledge.

The ongoing revolution in big data and statistical analysis is enabling the possibility of accelerating and standardizing fossil characterization and identification with machine learning techniques. In deep learning, machine learning models consisting of more than one artificial neural network layer, have the ability to learn representations of data with different levels of abstraction (LeCun et al. 2015). Recent advances in the architecture of deep learning convolutional neural networks (CNN) have greatly improved the fields of image classification and computer vision. LeCun et al. (2015) provided details on deep learning and showed some of the breakthoughs achieved by such technology. Dumoulin and Visin (2016) showed details on convolutions and arithmetic for deep learning procedures. We provide in the Appendix 1 the very basics of artificial neural networks and CNNs. Our material is built on images and examples to provide the reader with an intuitive understanding of the mechanics of deep learning and CNNs without delving into the details of the mathematical computations.

CNN models increase the levels of accuracy in numerous image classification tasks. For example, current CNN models can differentiate not only the image of a leopard from that of a mite or a container ship (objects with significantly different characteristics), but can differentiate images of leopards from their biological cousins –jaguars, cheetahs and snow leopards (objects with very similar characteristics; e.g. Krizhevsky et al., 2012). Szegedy et al.’s (2015) Inception V3 CNN architecture reached a 3.5% top-5 error (frequency in which the model cannot predict the correct class as one of the top five most probable guesses) and 17.3% top-1 error in the classification of the ImageNet Large-Scale Visual Recognition Challenge (Russakovsky et al.

2015). The training data for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), is a subset of ImageNet containing the 1000 mixed-object categories and 1.2 million images.

A currently underutilized application of machine learning is fossil identification - a key component of biostratigraphy. Ranaweera et al. (2009) used a computer-aided approach in which they applied clustering techniques followed by expert labeling for identification of foraminifers. Recently work has been done with the objective to generate a foraminiferal identification pipeline that uses CNN and other machine learning methods and compares such results to classifications performed by human experts (Zhong et al., 2017). Kong et al. (2016) show a novel technique that can be used for fossil pollen identification, by our understanding, the first application of CNN image classification techniques applied to fossil specimens. Kong et al. (2016) select patches of fossil pollen grains in microscopy images and use a pretrained CNN model to extract features for pollen species identification. Pollen researchers have been working on automated identification for a long time and are currently experimenting with CNN models as well (e.g. Sevillano and Aznarte 2018). Given the current proliferation of efforts to digitize biological specimens, both modern and fossil, (e.g. Blagoderov et al. 2012; Ellwood et al. 2015), successful application of CNN methods could greatly facilitate research that relies upon fossil identification and biostratigraphy.

We provide, to our knowledge, a novel attempt to conduct automated fossil classification using CNN models, and the first attempt on a fossil group (late Paleozoic fusulinids) identified specifically through 2D thin-section analysis. This methodology does not depend on specialized bench work and can be applied to existing photomicrographs in legacy collections –indirectly capturing the knowledge of the researchers that performed the classification or labeling of these collections. Our test case analyses provide proof-of-concept verification as highly accurate results were obtained with significantly smaller domain-specific training data when compared to traditional CNN applications. Although researchers are working with CNN models to perform image recognition using only a few examples, sometimes a single example per class is used for training (e. g. Koch 2015; Lake et al. 2015; Santoro et al. 2016), most of the CNN applications use hundreds to hundreds of thousands of examples per class. With the additional imaging of the numerous specimens in the large legacy collections of fusulinids in North America, automated classification can potentially organize a large volume of taxonomic and biostratigraphic information into a reliable and coherent system easily accessible to a variety of users, including both specialists and non-specialists. Our methodology uses data coming from traditional paleontology field and laboratory work and is dependent on specimen quality, but it does not aim to diminish the importance of current paleontology techniques. Our objective here is not only to help accelerate and spread fossil classification knowledge, but also to make fossil classification accessible to research groups with appropriate data, but little or no access to expert paleontologists.

Short Glossary

Because machine learning, and CNN in particular, may be unfamiliar to many paleontologists, we provide this simple glossary to define some of the technical terms used in the manuscript. More detailed machine learning definitions can be found in the list of references as well as online under "Machine Learning Glossary | Google Developers".

***Accuracy* –**The ratio between the number of correct classifications and the total number of classifications performed. Values range from 0.0 to 1.0 (equivalently, 0% to 100%). A perfect score of 1.0 means all classifications were correct whereas a score of 0.0 means all classifications were incorrect.

Class – The name, or category assigned to each data sample. In this paper we use “class” in the machine learning sense rather than in the biological sense.

Classification – The process of assigning data to a particular class.

Convolution **–** A mathematical operation that uses two functions, one generally interpreted as the “input”, and the other as a “filter”. The filter is commonly referred to as the “kernel”. The kernel is applied on the input, producing an output image or signal. In machine learning applications, a convolutional layer uses the convolutional kernel and the input data to train the convolutional kernel weights.

Cross entropy loss – a measure of the difference between the model’s predictions are from the provided label. Specifically, cross-entropy measures the difference between two probability distributions.

Convolutional Neural Network (***CNN***)– A neural network architecture in which at least one layer is a convolutional layer. Some authors also use ConvNets as a shorter term.

Deep neural network – An artificial neural network that uses more than one hidden layer. The process of using deep neural networks is sometimes referred to as deep learning.

Epoch – Generally used to depict a single pass through the full training set during the training stage. Not to be confused with a geological time epoch.

Fine Tuning – Execute a secondary training to further adjust the weights of an already trained model so the model can better achieve a secondary task.

Hyperparameter – The available “options” a user can change for different attempts to train a model. Hyperparameters contrast with weights/parameters that are automatically updated by the model, following the model's algorithm. For example, the number of epochs used to train a model a hyperparameter.

Labels **–** Names applied to an instance, sample, or example (for image classification, an image) associating it with a given class. In this paper the labels are the names of the target genus analyzed.

Loss **–** A measure of the model's performance, or how far the predictions are from the desired output.

Machine learning **–** a collection of approaches in which models improve their performance through automatic analysis of data.

Natural Images **–** A term commonly used in computer vision literature and without a strict definition. In a broad sense, the resulting color photograph taken with an ordinary camera.

Pooling **–** A filter that reduces the size of the input data, for example, replacing the value of four adjacent pixels with its maximum or mean.

Softmax **–** A function that provides the probability a sample belongs to each possible class.

Test data, test set **–** Samples not used in training but held aside to test the performance of the trained model. Ideally, the test set is used to evaluate only the final model, unlike the validation set that can be used to tune the model during training.

Training **–** An iterative process that determines the ideal parameters of a machine learning model.

Training data, training set **–** The subset of the data used for training.

***Transfer Learning* –** A technique that uses information learned in a primary machine learning task (e.g. bird classification) to perform a secondary machine learning task (e.g. fossil classification).

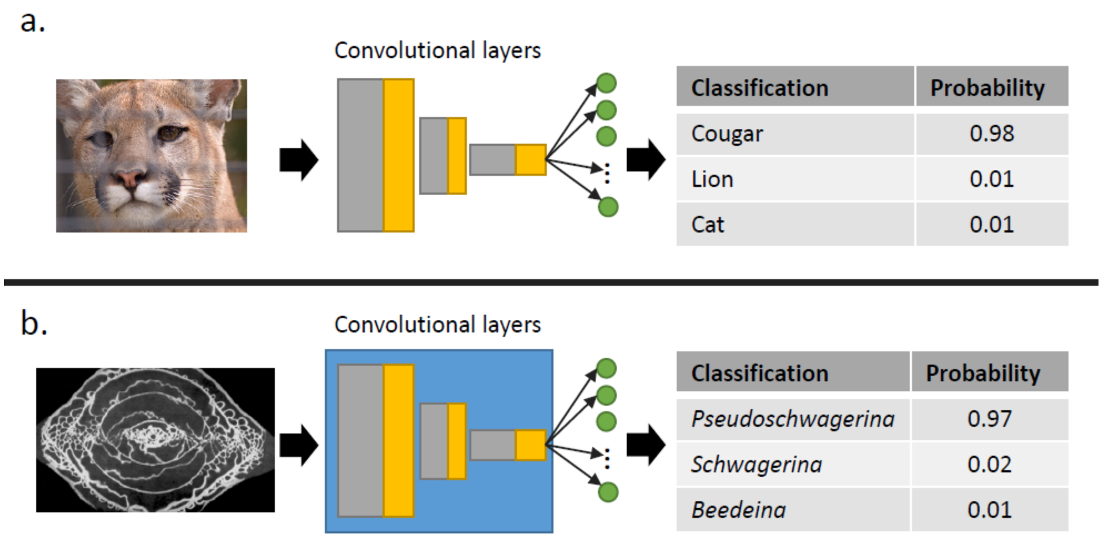
***Validation data, validation set* –** The subset of the data used to evaluate the training model during model construction.

***Weights/parameters* –** The coefficients of a machine learning model. In a simple linear equation, the slope and intercept are the weights of the model. In CNNs, the weights are the convolutional kernel values. The training objective is to find the ideal weights of the machine learning model.

Methods

In the realm of machine learning techniques, the task investigated can generally be divided into unsupervised or supervised learning. In unsupervised learning, the user provides data to the algorithm and the algorithm tries to identify patterns present in the data. In supervised learning, the user provides data and corresponding labels and the algorithm tries to learn a function or a relationship to map the data to the labels. In this paper we use supervised learning problem - we provide data (thin-section images) and labels for training and expect the CNN to provide a relationship between the data and the labels (the expert defined fusulinid genus or species).

In general, the reliability of CNN results is directly related to with the amount of labeled data used during training. With more examples provided to the CNN, the weights used by the model are improved, generating higher-accuracy and more reliable results. The CNN needs examples to recognize the features of each class it tries to differentiate. The work here focuses on assembling fusulinid thin-section data, and using transfer learning (Pan and Yang, 2010) to generate a CNN model to classify fusulinids. Figure 23 shows a simple representation of the transfer learning process. We provide more details of transfer learning in the next subsection.



Classification	Probability
Cougar	0.98
Lion	0.01
Cat	0.01

Classification	Probability
Pseudoschwagerina	0.97
Schwagerina	0.02
Beedeina	0.01

Figure 23: Visual representation of the transfer learning process. A CNN is trained on the primary task, generally containing many (millions) of samples. We generically represent convolutional and pooling layers with gray and golden rectangles whereas green circles represent densely connected neurons, commonly used in the classification layers. “Primary task” (a.) in this case represents an image from the ImageNet dataset going through a generic CNN model (convolutional layers and classification layers) trained on the same dataset. The CNN model then outputs the probability of the image belonging to one of the thousands of classes of the ImageNet. For “secondary task” we use the weights learned by the convolutional layers on primary task using the blue rectangle to represent weights learned on the primary task. We then train a new classification model.

Accurate identification of a fusulinid relies on attributes observable from an oriented section exposed along the long axis of the (prolate spheroid-shaped) fusulinid, bisecting the center. A transverse section is useful, but the longitudinal section is essential (Figure 24). Both sections reduce the complex internal morphology of fusulinids to two-dimensional views that can be easily imaged. Because fusulinid workers have used these oriented sections for years, an extremely large number of specimens oriented in the same manner exist in legacy collections in museums, although access to such images is not always easy. Thin-section collections, however, commonly consist not only of individual specimens of well-oriented longitudinal sections, but also thin-sections of fossil-bearing rocks in which cuts through specimens are randomly oriented and thus yield apparently different sizes and shapes. In this initial work, the training set contained only those thin-sections with well-oriented longitudinal cuts.

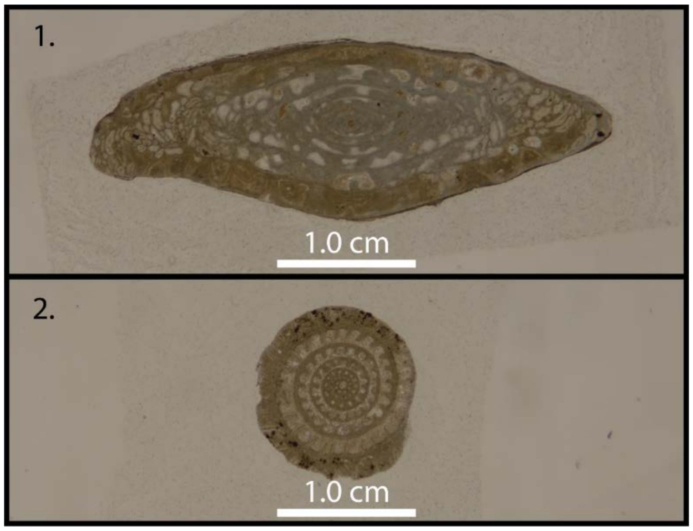


Figure 24: Thin-sections with different orientations from the analyzed collection: 1. *Beedeina mutabilis* with a longitudinal cut, and 2. *B. mutabilis* with a transverse cut.

Our fusulinid dataset comprises original fusulinid thin-sections from Waddell (1966) housed at the Sam Noble Museum at the University of Oklahoma (OU) imaged through modern digital photography. The Waddell collection comprises four different Pennsylvanian fusulinid genera: *Beedeina* (*Fusulina)*, *Wedekindellina*, *Triticites*, and *Fusulinella*. Samples from the American Museum of Natural History (AMNH) acquired through the iDigBio portal, an important initiative in digital access to biological collections, provided three additional Permian genera: *Parafusulina*, *Pseudofusulina*, and *Schwagerina*. Figures from Thompson (1954) and Walhman (2019) provide additional samples of *Beedeina* (*Fusulina*), *Fusulinella*, *Pseudofusulina*, *Pseudoschwagerina*, *Schwagerina*, *Wedekindellina*, and *Triticites*. We also extracted data from Williams (1963), Stevens and Stone (2009), Kobayashi (2012), Kossovaya et al. (2016) Kobayashi and Furutani (2019) Barrick and Wahlman (2019), and Wahlman (2019). Differences in thin-section image properties (e.g., background color), and data quality increased the difficulties encountered for training. Table 5 shows the data sources and Table 6 shows the data available for the fusulinid experiment. More details on the data used are provided in the supplemental material.

Table 5: Genus and source for the images used in this experiment.

Genus	**Source**
Beedeina	Alexander (1954), Waddell (1966), Barrick and Wahlman (2019), Wahlman (2019)
Fusulinella	Waddell (1966), Wahlman (2019)
Parafusulina	iDigBio, Stevens and Stone (2009), Kobayashi (2012),
Pseudofusulina	iDigBio, Kossovaya et al. (2016), Kobayashi and Furutani (2019), Thompson (1954)
Pseudoschwagerina	Thompson (1954), Williams (1963), Wahlman (2019),
Schwagerina	iDigBio, Williams (1963), Stevens and Stone (2009), Wahlman (2019)
Triticites	iDigBio, Williams (1963), Waddell (1966), Kobayashi and Furutani (2019), Wahlman (2019),
Wedekindellina	Waddell (1966), Barrick and Wahlman (2019), Wahlman (2019)

Table 6: Number of samples per class in each set. Note "class" here is used in the machine learning not the biological sense.

Class	**Training**	**Validation**	**Test**	**Total**
Beedeina	61	9	18	88
Fusulinella	10	2	3	15
Parafusulina	14	3	5	22
Pseudofusulina	28	5	9	42
Pseudoschwagerina	17	3	5	25
Schwagerina	42	6	12	60
Triticites	49	7	14	70
Wedekindellina	14	2	4	20

Transfer learning and data augmentation

Transfer learning can be used to address the shortage of sufficient domain-specific training data (Carranza-Rojas et al. 2017). In transfer learning, the learned characteristics of a base model trained on a primary dataset and task are reused for a secondary task (Yosinski et al. 2014). Therefore, layers previously trained with a substantial volume of labeled data can be used to address different objectives. Thus, a CNN model trained to identify the images of the ILSVRC can be used to classify fusulinid thin-sections with the help of transfer learning (Figure 23). In a study analyzing medical image data, Tajbakhsh et al. (2016) found that using transfer learning achieved results comparable to, or better than, training a CNN model from scratch (with randomly initialized weights). Yosinski et al. (2014) concluded that using transfer learning on subsets of ILSVRC classes perform better than training CNN models from scratch. Examples of transfer learning include Carranza-Rojas et al. (2017) for herbarium specimens, Esteva et al.

(2017) for skin cancer classification, Gomez Villa et al. (2017) for camera-trap images, and Pires de Lima et al. (2019a) for lithofacies classification. Pires de Lima et al. (2019b) showed examples of transfer learning using different geological images.

Deep neural networks have a cascading pattern in which the output of one processing layer is used as input to the next layer of the model. When trained on datasets of natural images, the first layers of CNN models learn features that resemble either color blobs or some variation of textures. This behavior is so common in CNN models that the analysis is reevaluated every time the initial layers learn any other image characteristics (other than colors or texture) resulting in a transition from general to specific features learned by the model (Yosinski et al. 2014). This behavior is why CNN with good performance on the ILSVRC (e.g. Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2014, 2015) can be successfully retrained for new, field-specific classification problems (e.g. Carranza-Rojas et al., 2017; Esteva et al., 2017; Gomez Villa et al., 2017; Norouzzadeh et al., 2018). As the layers become more specific the deeper they are in the model (i.e., closer to the output of the CNN than the input), some workers find it useful to extract only these more general image features. Kong et al. (2016) studied where the features of an ILSVRC pretrained model can be used without modifications to extract features from pollen data. Another powerful approach is to fine tune the ILSVRC model weights updating them with training data from the secondary task. Here, we apply three training modes: feature extraction, fine tune, and randomly initialized weights. Feature extraction “locks” (or “freezes”) the pre-trained layers extracted from the primary models. Fine tuning starts as feature extraction, with the primary model frozen, but eventually allows the all the layers of the model to learn. Randomly initialized weights mode starts the entire model with randomly initialized weights and all the weights are updated during training. Randomly initialized weights is not a transfer learning process, just the ordinary training. For the sake of standardization, all modes train the model for 100 epochs. In fine tuning, where part of the model is frozen, we first train for 50 epochs. Then we allow all layers of the model to be free to learn for another 50 epochs. We use five different well known CNN models: VGG19 (Simonyan and Zisserman, 2014), InceptionV3 (Szegedy et al., 2015), MobileNetV2 (Sandler et al., 2018), ResNet50 (He et al., 2016), and DenseNet121 (Huang et al., 2016) originally trained on ILSVRC. We use complete CNN models, substituting their last fully connected classification layers with our "top model". The top model is composed of an average pooling, followed by one fully connected layer with 512 neurons, a dropout layer (Srivastava et al., 2014) used during training, and a final fully connected layer with a softmax output where the number of neurons is dependent of the number of classes for the task (eight genus, eight neurons). The models are trained using Keras (Chollet et al. 2015), with TensorFlow as its backend (Abadi et al., 2016). When kernels are initialized, we use the Glorot uniform (Glorot and Bengio, 2010) distribution of weights. The models are optimized using a stochastic gradient descent with a learning rate of 1.0e-3 and momentum of 0.9 to minimize the categorical cross entropy loss. Rather than trying to find the best accuracy possible, our objective is to show the behavior of different CNN models and training modes using thin-section fusulinid data. Therefore, we choose to keep the hyperparameters fixed, as described for all experiments. We use a NVIDIA GeForce RTX 2060 for the experiments. Even though transfer learning provides a powerful approach to address the problem of an insufficient amount of training data, and has been successfully implemented in different fields, the relatively small number of digitized thin-sections available for this work created challenges in assembling the training set. Recent examples using transfer learning for image classification employed training datasets of 10^5 images (Esteva et al., 2017; Gomez Villa, et al. 2017; Carranza-Rojas et al., 2017). In contrast, we relied upon 10^2 original images of fusulinid specimens, three orders of magnitude smaller than other studies using transfer learning.

Owing to this limited dataset for training the CNN, we used a common bootstrap process to generate pseudo-samples using the available images. The population was augmented by simple data rotation. Each longitudinally aligned original image was rotated through a range of angles ±5°about the horizontal axis, as well as flipped about the horizontal and vertical axes to expand the training data set. Such approaches increase the number of images that could be used for training and help in the generalization of the model. Therefore, to facilitate training and reduce the chance of overfitting, we augment the training and validation data using Bloice et al's. (2019) Augmentor tool, as well as Kera’s generators pipelines.

Results

We fit five different models using three different training modes, effectively performing 15 experiments. Figure 25 shows an example of the loss and accuracy evolution during fine tune training of InceptionV3. Plots like Figure 25 are useful to investigate whether or not the models are overfitting the training data. Because of the many parameters CNN usually have, it is possible for the model to simply remember all the training data and have a poor generalization performance, i.e. poor performance when the model classifies new data. Ideally, the training and validation set curves should be close to each other, although in the great majority of cases the models have a better performance on the training set than on the validation set. Figure 25 shows that the model starts to overfit in the first training stage, when part of the model is frozen, and validation accuracy only starts to increase on the second stage of fine tune training, when all the layers in the model are allowed to change their weights. The results in Figure 25 show that the model is in fact overfitting the training set. The accuracy for the training set is 1.0 on the final epochs and the loss is very close to zero, whereas validation set metrics are not perfect. With the current implementation, more data would help prevent overfitting.

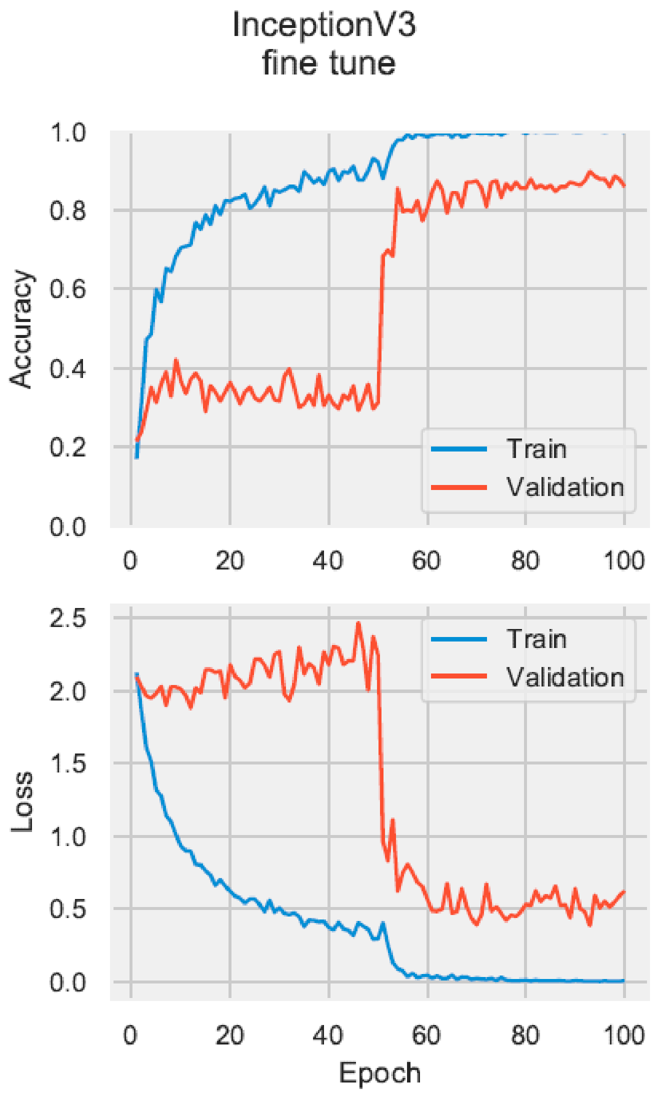


Figure 25: Loss and accuracy evolution of train and validation set during training InceptionV3 in the fine tune mode.

Because the validation set is used during training, a better evaluation of the model's generalization is obtained using the test set. The test set is never used during training and provides the expected performance of the model on data of the same quality, but that were never seen by the model. Figure 26 shows the test set accuracy of all the models and training modes analyzed. Finer details for the best performances for each one of the models are presented in Table 7. Although the models are overfitting the training data, their performance on the validation and test set is appropriate, producing high levels of accuracy.

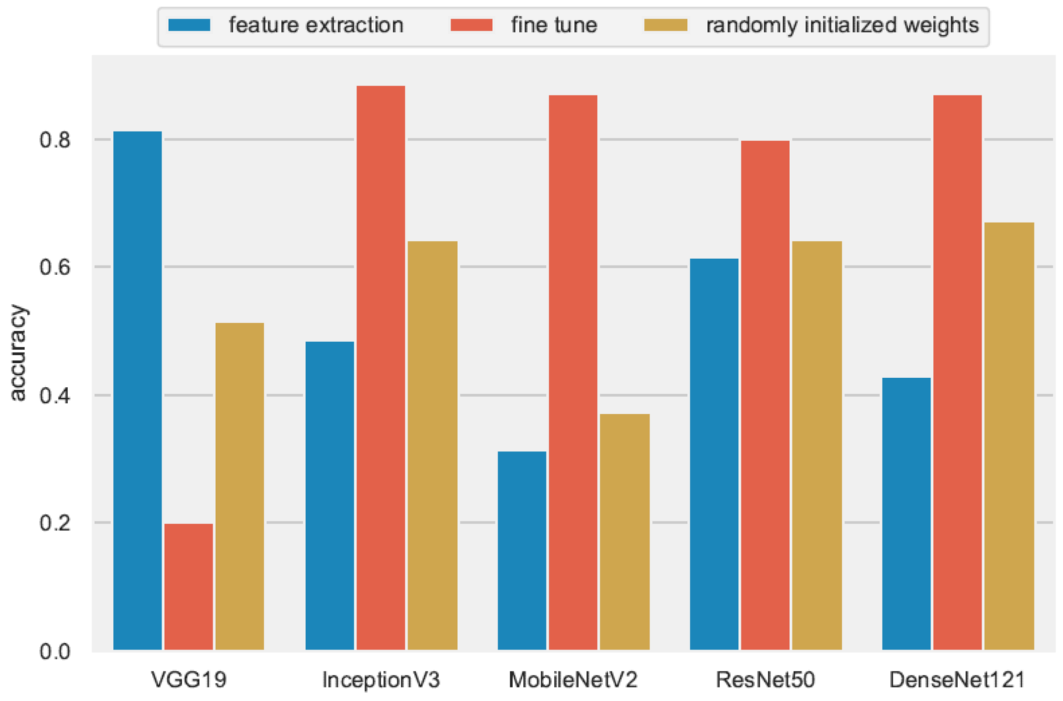


Figure 26: Test set accuracy for the five different models using three different training modes.

Figure 27 shows the confusion matrix computed on the test set using InceptionV3 trained on the fine tune mode. Confusion matrices are helpful to summarize the differences between the classifications performed by the model and the classification performed by the paleontologists. Figure 28 shows examples of fine tuned InceptionV3 classification of images in the test set and provide more details of the strengths and weaknesses of the mode. The loss and accuracy evolution during training, the confusion matrix, and the classification of the test set of all models and training modes are available in the supplemental material.

Table 7: Accuracy for the highest performing training mode for each one of the models.

Model	Mode	Accuracy
VGG19	feature extraction	0.81
InceptionV3	fine tuning	0.89
MobileNetV2	fine tuning	0.87
ResNet50	fine tuning	0.80
DenseNet121	fine tuning	0.87

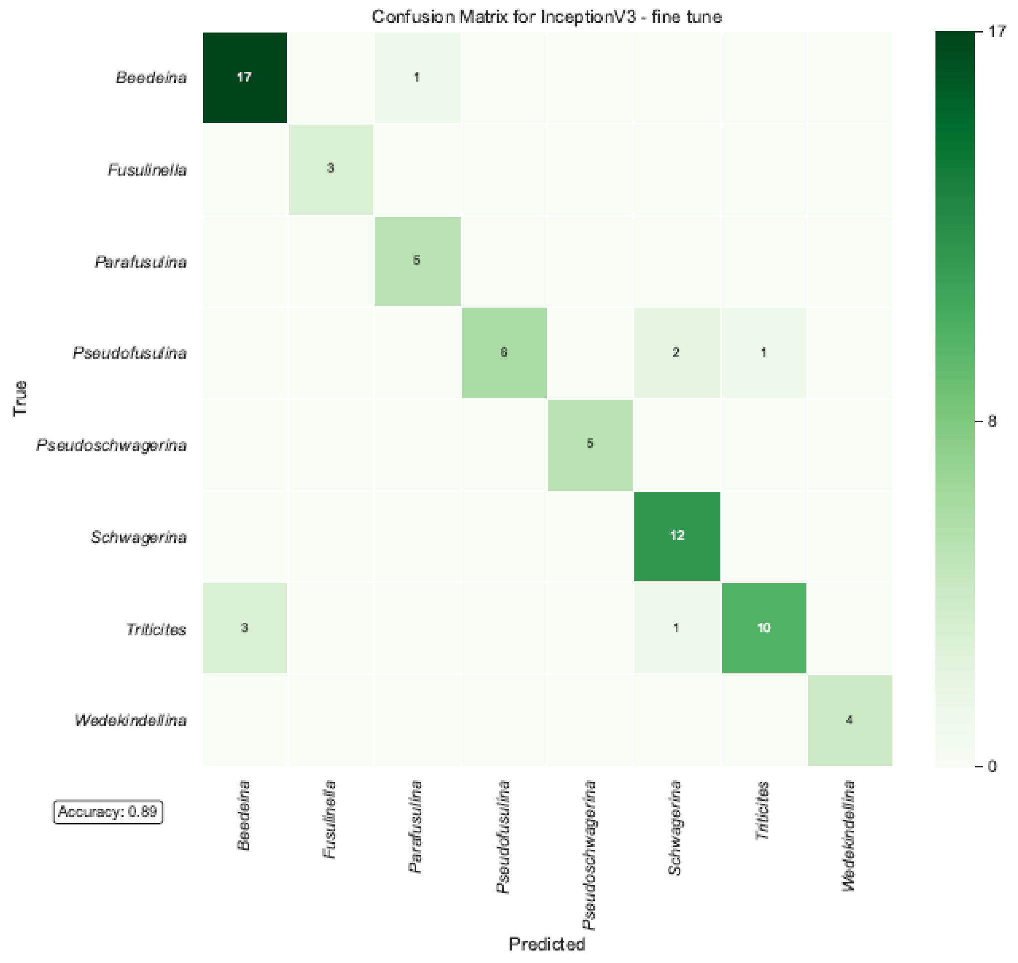


Figure 27: Confusion matrix for InceptionV3 trained in the fine tune mode. The confusion matrix shows the expert provided labels vs the model predicted labels. A perfect agreement between model and expert yields a matrix with values only on the main diagonal. Zero values are omitted.

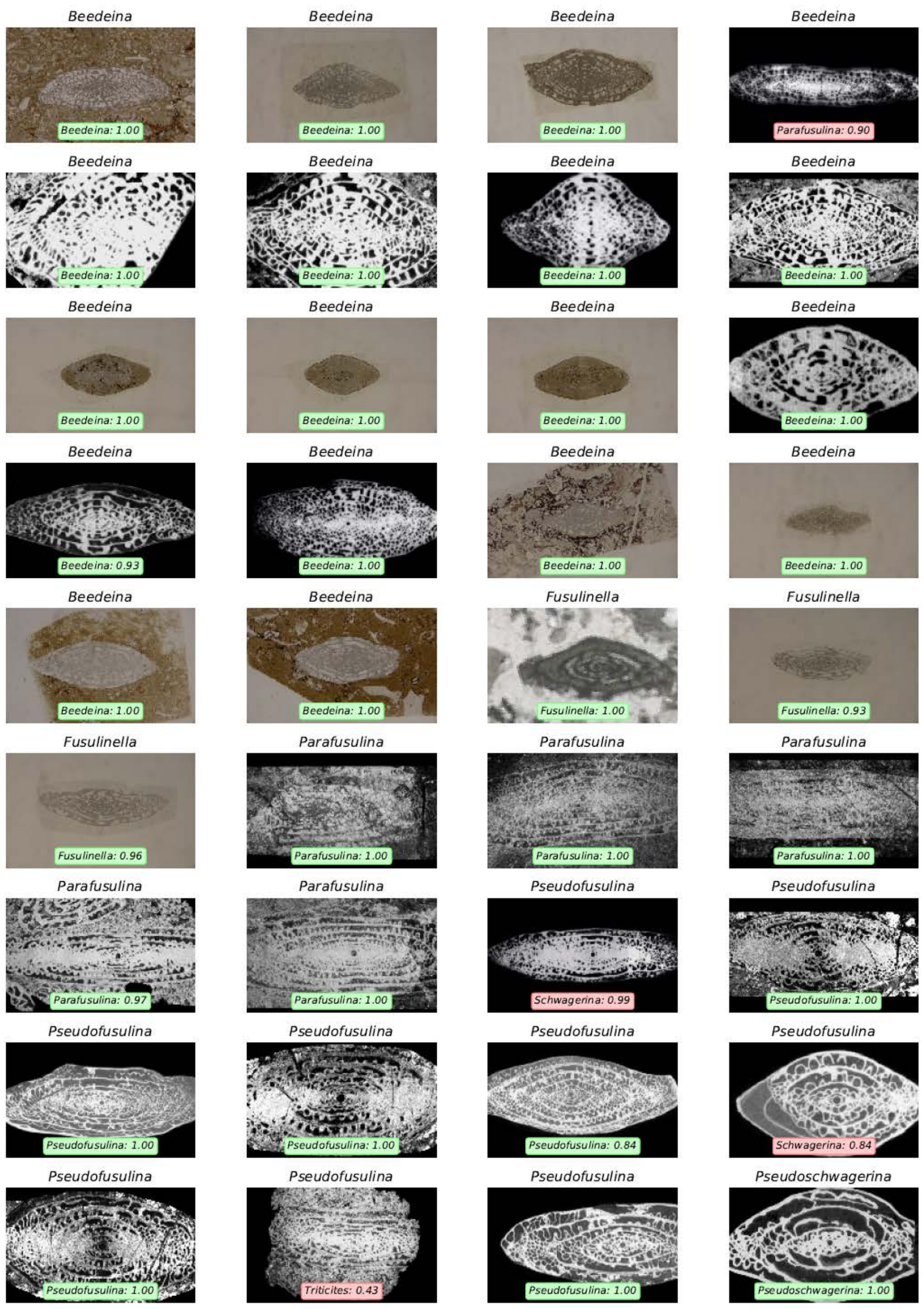


Figure 28: Examples of images in the test set classified by InceptionV3 trained on the fine tuning mode. The titles of each of the images are the classification provided by the paleontologists in their original publication, while the text boxes inside the thin-section images are the classification provided by the CNN model. The text box is green when the model assigned the same class (biological genus) as the paleontologist, and red otherwise. The value in the text box shows the probability assigned by the CNN model for that class.

Discussion

Despite the difference between the classification of ILSVRC's natural images and the thin-section fusulinid classification task, the CNN models trained on ILSVRC learned to extract features that are useful for fossil classification. Figure 26 and Table 7 show how different training modes can affect the model's performance and how fine tuning a model previously trained on a large and complex dataset such as the ILSVRC outperformed other training modes. Loss and accuracy for training and validation figures in the supplemental material show that, in general, the feature extraction training mode seems to overfit the data very quickly. Randomly initialized weights validation metrics apparently starts plateauing with a high level of overfitting as well. Thus, updating the initial ILSVRC-based weights on the CNN models with fine tuning is more effective than training a model with randomly initialized weights, except for VGG19 in our case. As the ILSVRC is a complex task in which samples in the same class are very different from each other, the models need to learn very effective transformations to properly differentiate the classes. Because thin-sections have specific characteristics that differ from more common natural images, such transformations need to be updated for a proper thin-section classification. Curiously, fine tuning VGG19 with our choice of hyperparameters led the model to a local minimum and degraded highly accurate results found using feature extraction for such a model. It is likely that both a hyperparameter search or more samples could help prevent degrading of results for the model, but such analysis is beyond the scope of this study.

To our knowledge this is the first paper providing details on the use of thin-sections, commonly used in fusulinid biostratigraphy, as input for a CNN model that can be used to identify microfossils. Our dataset comprises thin-section images from different sources and with different quality. In the approach we use here, the final user can simply provide an adequate image and the CNN model would output the probability of assignment of the specimen to a fusulinid genus – given that the model can only choose from the classes (biological genus in this case) on which it was trained. This study differs from Kong et al. (2016) because we use 2D thin-section images (not 3D stacked images), and our process uses the complete image during training and testing (not selected patches). Biological complications aside, we also achieved similar accuracy (87%) in our best performing results, differentiating more classes (eight genera) than Kong et al. (2016) (three species). Even lacking an extensive database of images, the methodology we applied achieved high levels of accuracy. Although we present only classification of genera, we are certain that classification of fusulinids to species using the methods we propose is possible. Groves and Reisdorph (2009) used multivariate morphometry to show that the *Beedeina* species separation is statistically significant. With appropriate data and more samples per species, our CNN methodology should able to achieve high levels of accuracy because these are clearly morphologically distinct species.

Unlike a human interpreter who relies upon a defined set of morphological measurements to perform taxonomic classifications, the CNN operates from no knowledge of specific attribute analysis and performs the classification based on image characteristics alone. This observation also implies that a CNN model, at least with this current implementation, cannot be used to define a new taxonomic division (e.g., a new species), although it may help separate out specimens that do fit into existing species. The set of transformations created by the CNN are abstract and do not rely on specific phylogenetic systematics measurements; rather, the rules are akin to a cascading set of filters. These filters perhaps are generating rules that approximate their behavior to the measurements used during taxonomy studies. But because the CNN models have many such filters, it is often difficult to discuss the interpretability of CNN models. CNN interpretability by itself is a topic in research (e. g. Olah et al. 2017, 2018). When analyzing a new image, the CNN model, as implemented in this study, will always generate a set of probabilities that such image belongs to the CNN’s learned classes, never declaring the image is none of the pre-defined classes. Nonetheless, the methodology we implemented in this project can easily be generalized and will improve as new images are digitized and made available to the scientific community. Considering that different taxonomic divisions request different attribute analysis – e.g. during the interpretation of conodonts, specimen surface texture is not as important as caudal point and rostral point (e.g. Hogancamp and Manship 2016) – we envision that CNN techniques will go through more significant modifications as they are applied to other taxonomic groups.

Although our approach is similar to recent studies employing transfer learning in image classification (e.g. Carranza-Rojas et al., 2017; Esteva et al., 2017; Gomez Villa et al., 2017), the work we presented achieves highly reliable fossil classification using a limited domain-specific dataset three orders-of-magnitude smaller than used in these referenced studies. Kong et al. (2016) used data acquired with a fluorescence microscope with a form of structured illumination to produce high-resolution, 3D image stacks. In contrast, the data we used in this study was acquired through photomicrographs (2D) taken with an inexpensive and easily available consumer camera and lens, as well as many samples simply obtained by searching published literature. We assume preparation of the fusulinid samples in all original studies was done by standard methods that have been in use for decades by paleontologists. Because we used such standard image data, we predict the methodology used in this paper has potential for wide applicability and rapid deployment with minimal start-up costs. As more image data are digitized, the technique we use can be applied without the need for laboratory-specific tools and knowledge, which represents a significant improvement over previous approaches requiring specialty image acquisition for CNN (e.g., Zhong et al., 2017). In fact, many of our samples were simply extracted from online versions of published literature, making our dataset significantly more irregular.

As digitization of legacy data accelerates, the approach presented here will improve with more detailed image processing. Image segmentation techniques can be used to clip the thin-sections containing significant presence of biotic or abiotic components (noise) besides the organism being analyzed; this will help both in the CNN training and in sequential sample classification. With more data available, object detection, the computer vision task to detect occurrences of objects of different classes (Szegedy et al., 2013; Zhao et al., 2018; Agarwal et al., 2018), can be applied, increasing the potential of paleo-tailored CNN's in the identification of varying taxa captured in the same sample. The technique we demonstrated in this paper is very general and can easily be modified to suit the identification of different fossil groups, such as conodonts, ostracods, ammonites, and others, as long as the specimen can be classified with a 2-dimensional representation (thin-section or comparable digital image).

However, we acknowledge that CNNs may be harder to apply to other fossil groups. For example, in trilobites it is unusual to find a complete specimen; in most cases the membrane connecting the thorax, cephalon, and pygidium will deteriorate, causing the exoskeleton to fall apart. This leads to paleontologists having to make identifications using fragmented and isolated sclerites. Using trilobite sclerite images as a means of training and classification would likely not work as well as a dataset with complete specimens, such as those that can be obtained for other fossil groups.

As the CNN models are trained with expert labeled data, such expertise is captured in the model’s weights in the deep neural network. Therefore, a CNN model, trained on different collections and having input from different paleontologist experts, provides a means of sharing collections and interpretations across great distances. A fusulinid expert working in the US can help train a CNN fusulinid classifier and such a model (and an abstract form of the expert knowledge) can be used in Asia with no significant cost; in the meantime, researchers working with *Saccorhytus* fossils in China (Han et al., 2017) can train another CNN to classify their data. Such ease in the exchange of knowledge can help validate interpretations of data spread around the world.

If we are able to capture and mix different paleontological expertise (training CNNs to identify a wide range of taxa), such models can be helpful to identify specimens that might have previously been misclassified. The combination of CNN as an easy-to-use but highly accurate tool and the digitization of stored paleontological samples can provide a rapid method to bring collections residing in museum drawers for decades to light. With easy access to this valuable data, the community can then apply modern statistics to better analyze spatial and temporal distributions, construct more precise assemblages, or simply track evolutionary trends more effectively. We should not discount the “discovery” component; museums commonly have a special exhibition of a fossil or bone that was collected decades ago and was only now identified as a new genus/species.

Conclusion

The most evident drawback of the methodology we apply here is its current dependency on a large amount of data to generate robust classification models. The dataset we use was created with relatively few original images, even though it uses images from different sources. Notwithstanding, our tests show that CNNs can correctly identify fusulinid specimens to genus level with a significantly high probability when compared to the other taxonomic classification options. With access to more labeled data, training can be improved, enabling the generation of a model sufficiently robust to overcome complications such as the presence of more than one specimen, geologic noise, among others. The move towards digitization of biological and paleontological collections at numerous museums will provide the big data enhancement to enable assessment of the CNN methodology for examples of fossils from around the world, and ultimately identification to species level.

Efforts in data digitization are important initiatives to protect scientific knowledge and the approach documented here contributes to such endeavors, and aids the use of biostratigraphic data in the scientific community. Biological variation, differences in specimen size, different imaging techniques, and other considerations will complicate the automation of the classification process, but can ultimately lead to deeper understanding, and significant enhancement for all work that relies upon fossil identification.

Acknowledgements

We thank the iDigBio initiative for providing access to the community for biodiversity collections data. Rafael acknowledges CNPq (grant 203589/2014-9) for the financial support and CPRM for granting the leave of absence allowing the pursuit of his Ph.D. studies.

Appendix 1. Basics of deep convolutional neural networks

In this section we provide the reader with an overview of the essentials of image convolution, deep learning, and convolutional neural networks (CNNs). We avoid detailed mathematical explanation but rather build the fundamental intuition necessary for CNNs comprehension and usage through figures and examples. LeCun et al. (2015) provide a comprehensive description of deep learning, with information about the building blocks and techniques common to several artificial intelligence applications. Dumoulin and Visin (2016) showed details on convolutions and arithmetic for deep learning procedures. Krizhevsky et al.'s (2012) work is considered a breakthrough that used CNNs to reduce the error rate for object recognition tasks by almost 50%, leading to the subsequent adoption of deep learning in multiple fields of study.

Images and convolution

Images can be digitally stored in several different ways. Raster images (as opposed to vector graphics) are built upon the characteristics of single points (pixels - short for Picture Element). Each pixel has attributes that comprise the data necessary to represent that part of the picture. An image is a grid or matrix of such pixels.

Although there are different ways for the pixels to be archived, one of the most common methods of storage is to use different levels of red, green, and blue (RGB) to represent a wide range of colors, such that an image file then can be split into its three building bands or channels (Figure A. 1). Therefore, images are digitally stored as a 3D volume, defined by a height, a width, and a “depth” of the three channels. This matrix structure is amenable to diverse image processing techniques, such as convolutions, that are essential for the image recognition tasks performed by CNNs.

Convolutions are oftentimes considered to be the most important operation in signal processing. Convolutions are a linear operation in which an input is filtered by a function (or kernel) generating an output; such operations can be generalized to *n*-dimensional spaces. Convolutions are related to cross-variance and are used in different fields with slightly different definitions, all essentially depicting the same operation. Here, to continue developing the essential intuition of how CNNs operate for image recognition tasks, we use 2D convolutions applied to each color image, one at a time. In our images, the kernel slides from left-to-right then top-to-bottom.

Figure A. 2 shows how convolutions are performed using 2D matrices and a 3 by 3 kernel. The kernel slides through an input and generates an output. The computation performed at each step is a simple convolution, or sum of the values of the input aligned and weighted by the values of the kernel themselves. Special care needs to be taken for image borders. The stride of the computation can be every pixel, every second pixel or some other multiple (Dumoulin and Visin, 2016). Note the output in Figure A. 2 does not have the same size as the input. Kernel sizes other than 3 x 3 can be used with the same essential technique.

In Figure A. 3 we show the results of image convolutions. The convolution of a three-channel image with a three-channel kernel results in a three-channel output (CNN layers eventually sum the results of such convolutions collapsing them into a single channel output, thereby generating a single channel “grayscale” image). In the example we provide, the red kernel is set in a way in which the resulting output will highlight horizontal changes. The green and blue kernels are set as the identity, so there are no changes in the output when compared with the original bands. Several results can be obtained using these convolutions with different size kernels and different weights (the values for each one of the elements in the kernels). In

CNN applications, the values for each one of the kernels are randomly initialized and, during the training process, they are modified iteratively in a way that their outputs will help differentiate the analyzed classes. Figure A. 4 shows familiar examples of processed images that can be obtained with filters based on convolutions, most of which are available in common image processing software.

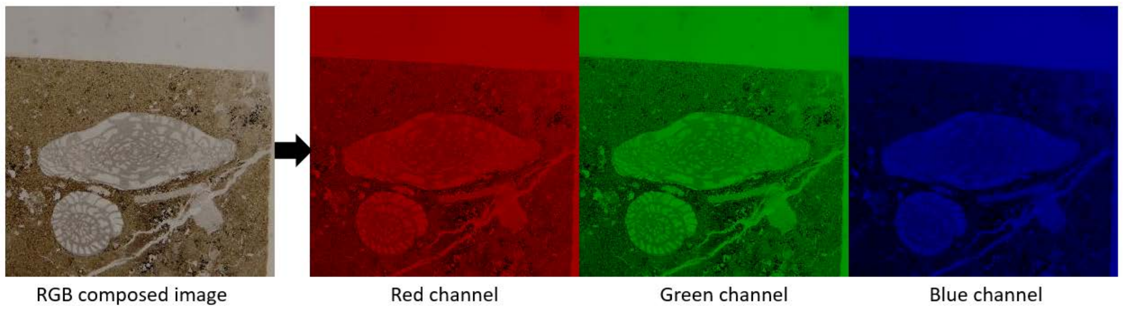


Figure A. 1: Decomposition of an image into its three RGB channels.

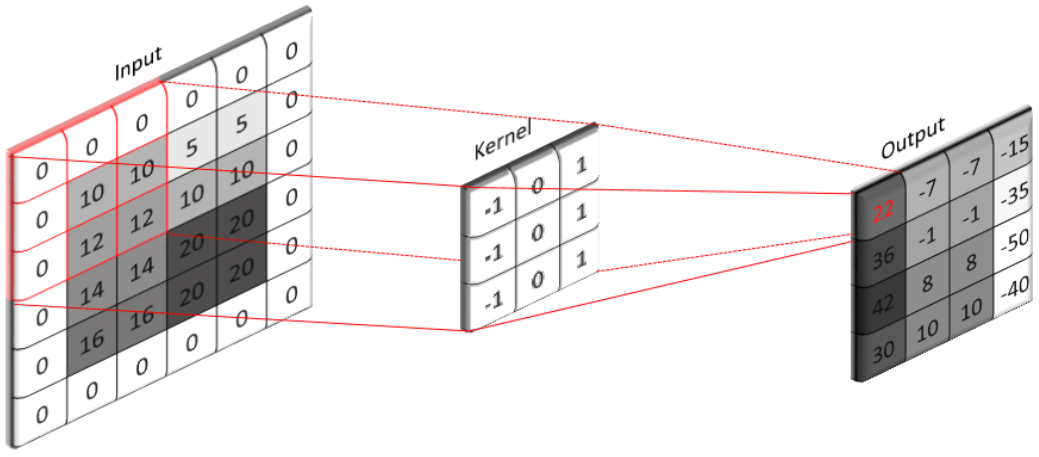


Figure A. 2: Visual representation of a 2D convolution. The input (left) is filtered by the kernel (center) generating the output (right). This operation can be visualized as a sliding window of operation; the kernel slides over the input computing the output as a dot product between the values of the input (seen by the kernel window) with the values of the kernel themselves. In our example, the borders of the input are all zeroes and the kernel slides with stride one; after the first value (22) is computed, the kernel slides one pixel to the right and performs the operation again. Different strides and padding strategies can be used to generate different outputs.

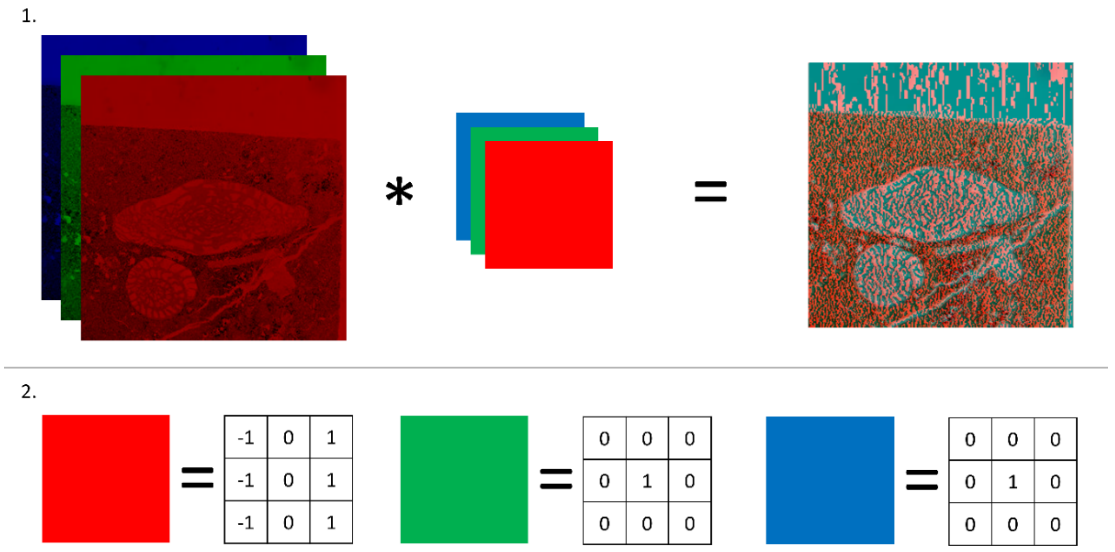


Figure A. 3: Convolution of images. Top panel (1.) represents the convolution of the input with RGB kernels and the resulting image. 2. Values of RGB kernels. Red channel in input is convolved with the red kernel, green channel with green kernel, blue channel with blue kernel. The resulting image shows the horizontal edges in the red channel, while the green and red components remain unchanged.

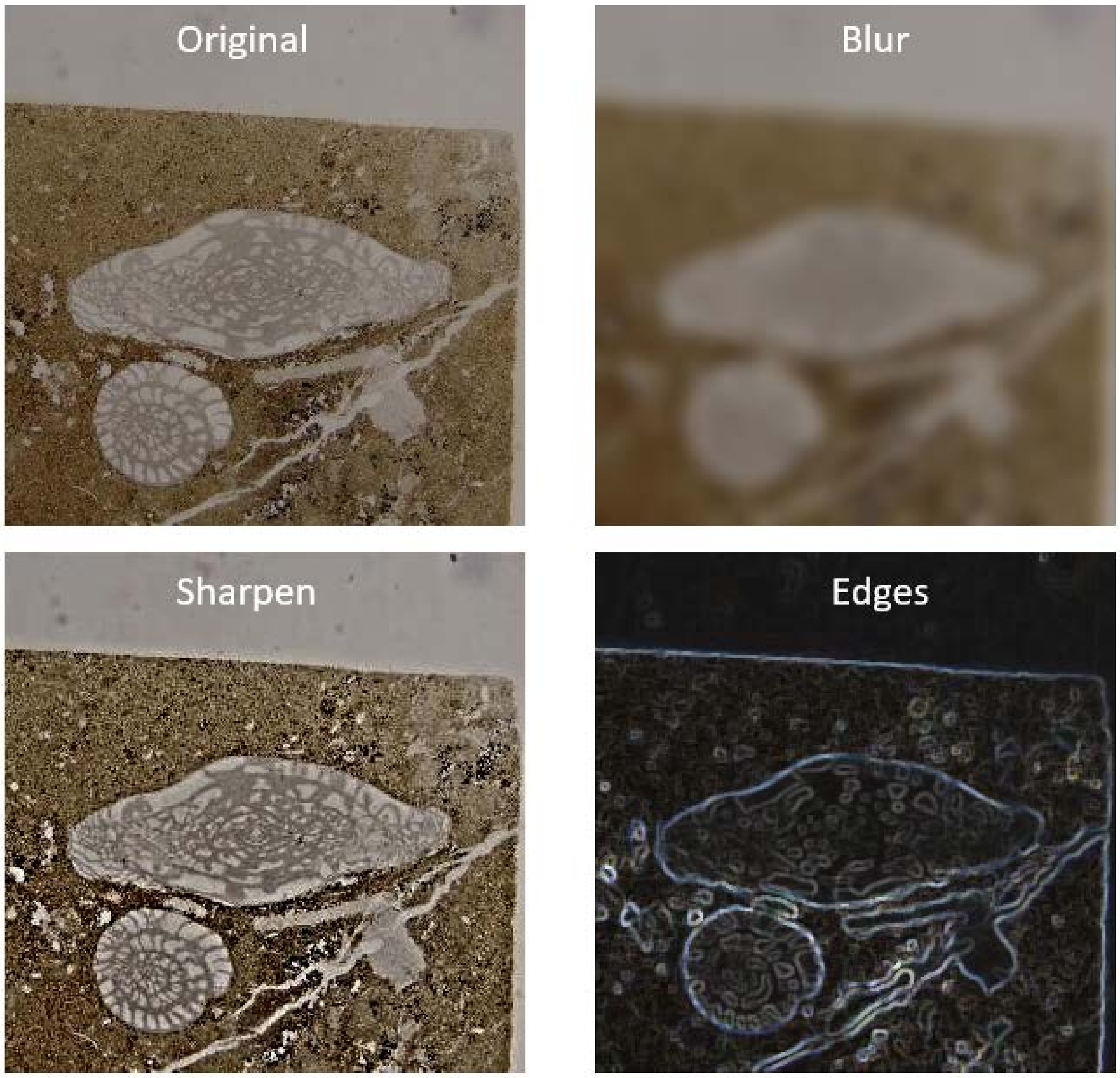


Figure A. 4: Original image and result of different image processing techniques. All the processed images are results of various types of convolutions.

Single neuron and an overview of artificial neural networks

A single artificial neuron is a linear transformation like those shown in Figure A. 4, followed by a nonlinear transformation, traditionally called the “activation function”. An artificial neural “network” then is composed of multiple neurons, each one with different weights.

The very first steps in artificial neural networks modeling are the same as linear fitting statistics frequently used in many fields. The refinement and improved power of artificial neural networks comes from the addition of the activation functions that modify the linear (e.g. convolutional) transformation to a nonlinear transformation. There are several activation functions traditionally used when building artificial neural networks. Functions like the sigmoid, hyperbolic tangent (tanh), rectified linear units (ReLu), and leaky ReLu are very common choices as activation functions (Figure A. 5). In CNNs, ReLu are very frequently used in the hidden layers whereas the sigmoid and softmax $S(\boldsymbol{x_i})$ given by

$$S(\boldsymbol{x_i}) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}$$

are used in the classification step (the last layer).

Figure A. 6 shows a visual representation of the computations performed in neural networks. A single neuron receives data from different inputs and processes such data generating an output. Each neuron has its own set of weights (these are the values that are modified during the training step) and an activation function that has its behavior independent of the input data. The combination of neurons receiving the same input is commonly called a layer. When more than one layer exists in a neural network, such layers are often called hidden layers. The layers of Figure A. 6b are “fully connected” having connections to all elements in the previous layers.

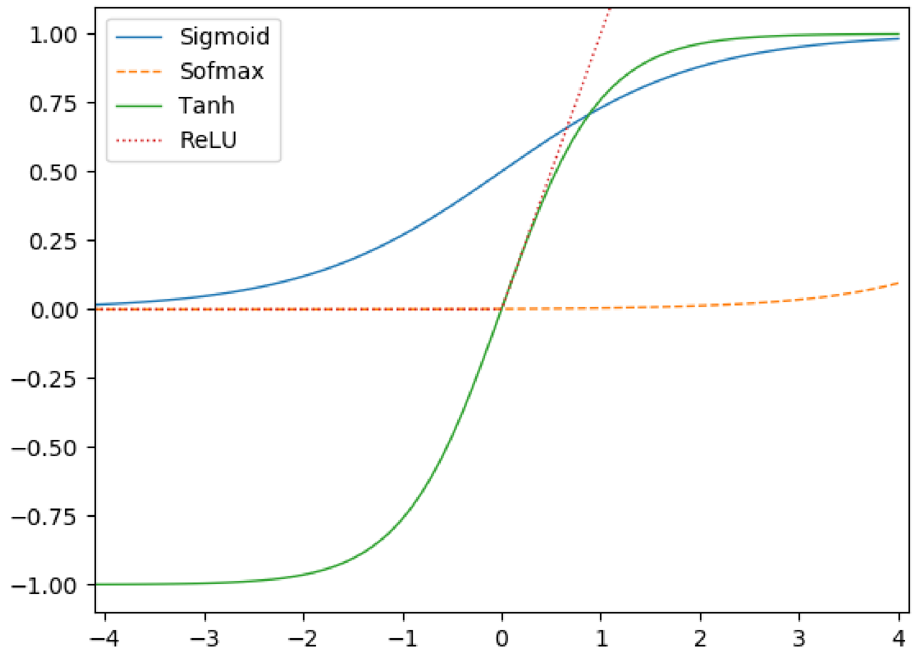


Figure A. 5: Examples of commonly used activation functions. Note the sum of softmax is 1, so its appearance varies according to the horizontal axis (range and sample rate).

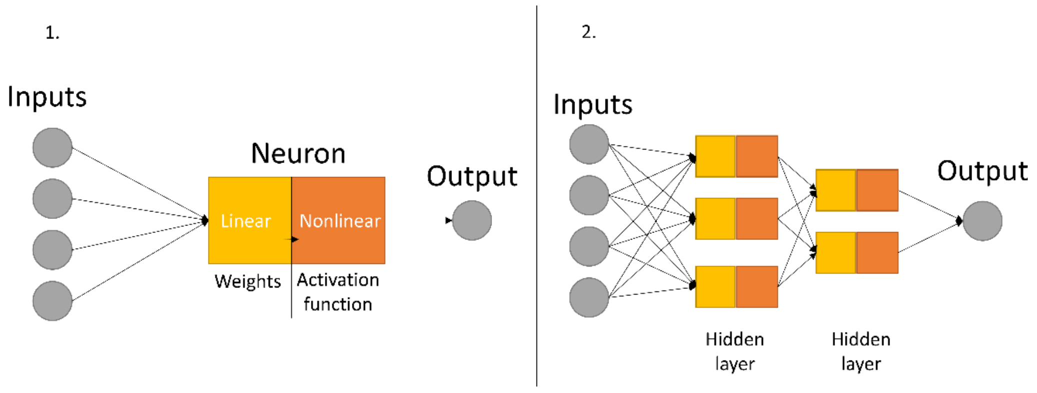


Figure A. 6: Visual representation of neuron (1.) and an artificial neural network (2.). The number of neurons and layers in 2. are arbitrary (as well as the number of inputs). Consider, for example, that each one of the circles in the input represents RGB colors for an image plus a bias term. Next, assume that three RGB images and a bias are sufficient to differentiate objects and assign them to classes 0 and 1. The neuron then iteratively modifies the values of the weights that when applied to the input and fed to the activation function successfully assigns the input to class 0 or to class 1.

(Deep) Convolutional neural networks

Given these simple principles of digital images, convolution, and neural networks, we now continue our intuitive analysis of how CNNs work. The prefix “deep” is sometimes used when referring to CNNs. Until recently, many neural networks have been built on handcrafted features, whereas now most of the feature transformation is performed by CNN models. In this section, we use the terms described before to show how an image changes when going through the layers of CNNs.

Figure A. 7 shows the flowchart of the operations described in this section. These layers (convolutions, pooling) are very common in CNNs. In the example we give, the convolution layer weights (the values for each one of the elements in the kernel) were set according to our choice. In CNN model training, these weights are initialized with random values and are iteratively updated. Interestingly, as observed by Yosinski et al. (2014), the first layers of a CNN will have weights that act as edge or color detection kernels – much like the ones we chose in this section. As the kernel values change according to the necessity of the CNN to reduce the error (difference between the predicted class and the true class), different transformations are applied to the images (such as in Figure A. 4).

Figure A. 8 shows a visual representation of the 3 x 3 convolution layer operations as well as the resulting image for the same fusulinid image we have been using in this section. The resulting image is then input to a pooling layer. Pooling layers reduce the height and width of images with simple statistics and are commonly used in CNNs. Pooling layers can have different sizes (height, width) and different strides. Figure A. 9 shows an example of max pooling, which captures the maximum value of a subset of the input for a small submatrix. In Figure A. 10 we show the resulting max pooling of the output of the 3 x 3 convolution (output of Figure A. 8) continuing the flowchart depicted in Figure A. 7.

We provided the elements necessary for an overall awareness of how CNNs operate and the results of a very small (and "incomplete") CNN. We believe this material can be used as a quick reference for some of the common terms used in deep learning techniques. Understanding how an image is transformed through convolutions and pooling can help demystify the results obtained by CNN applications. With the information given in this section, we believe the reader can quickly grasp the differences in architecture of robust CNN models, as well as imagine how the data are transformed as they go through the models.

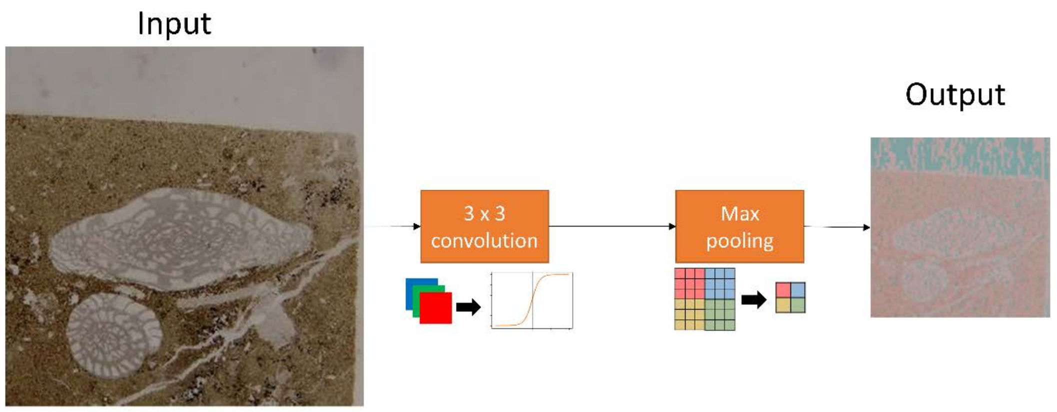


Figure A. 7: A simple flowchart representation of the operations performed with the input image. Note that the 3 x 3 convolution block also accounts for the activation function. In the max pooling process, the maximum value of a subset of the input is stored and the image's size (height and width) is reduced. Here we represent the resulting image as a composition of RGB channels which, during CNN applications such result will eventually be stacked into a single channel. Figure A. 9 shows how a max pooling layer works.

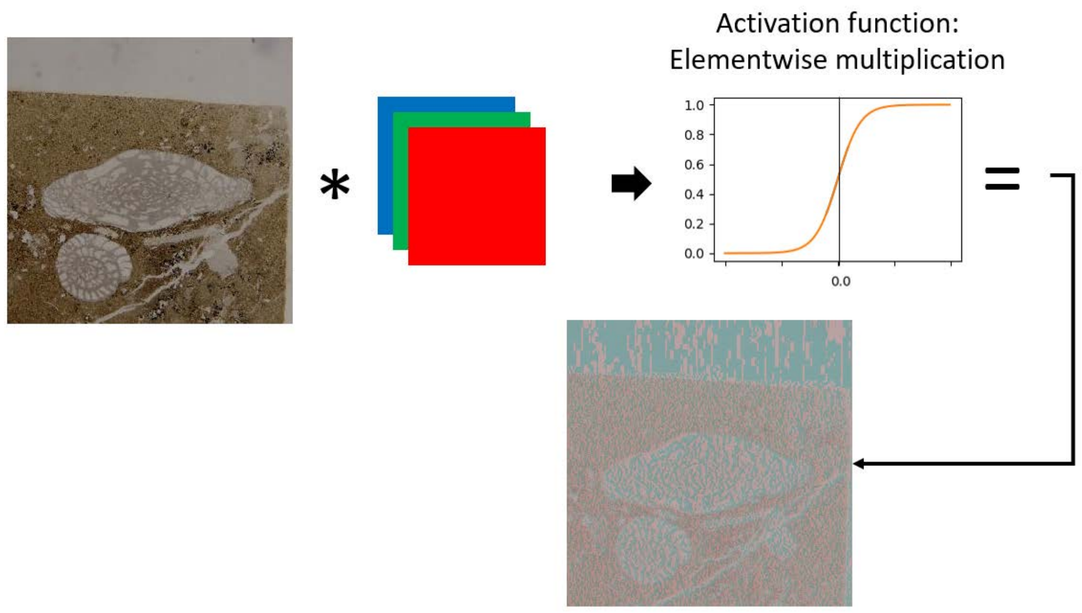


Figure A. 8: Visual representation of the image transformation occurring when an image goes through one convolution neuron. In this example, the image is convolved with the same kernel presented in Figure A. 3. The result of the convolution then goes through the activation function in which every pixel has its values scaled. Note how the resulting image after the activation function is similar to the one in Figure A. 3; this is a simple (but nonlinear) rescaling process. In general, the resulting process is a single channel (grayscale) image as the three bands are stacked, we choose to represent the fusulinid in RGB for better visualization.

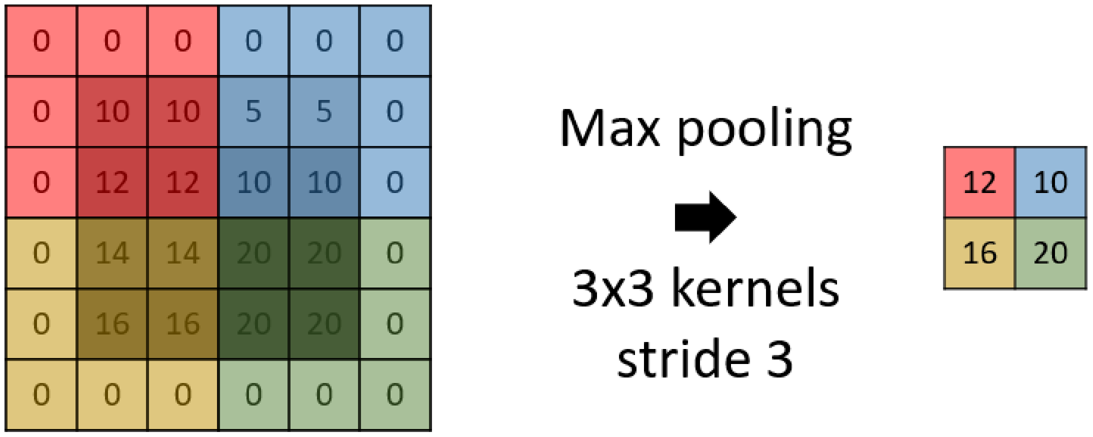


Figure A. 9: A representation of the “max pooling” operation, where the output is simply the maximum value of a subset of the input. Colors are used to facilitate visualization. Different strides and padding techniques can be used with pooling layers, as well as different statistics (minimum, average, median, or other statistical measures). In this example, the stride is two in each direction such that the windows do not overlap.

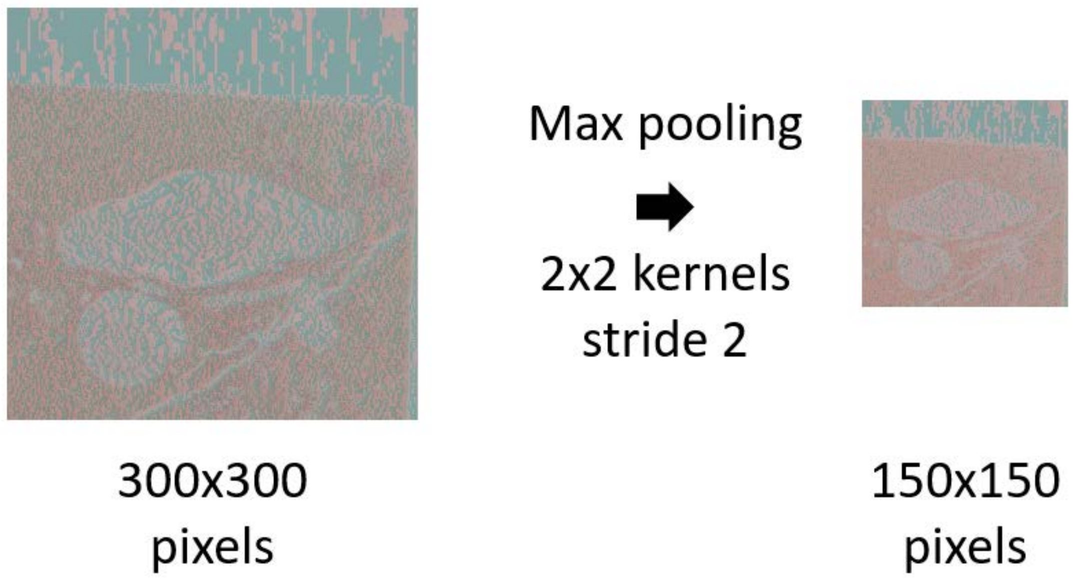


Figure A. 10: Max pooling of the image result shown in Figure A. 8. Note that the image is essentially the same, but that the colors are slightly different and that the image size (height and width) has been reduced.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. and Zheng, X. 2016. TensorFlow: A system for large-scale machine learning, p. 265–283. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). Retrieved from https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

Agarwal, S., Ogier Du Terrail, J. and Jurie, F. 2018. Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks. ArXiv e-prints.

Alexander, R. 1954. Desmoinesian Fusulinids of Northeastern Oklahoma. Oklahoma Geological Survey. Retrieved July 15, 2018, from http://ogs.ou.edu/docs/circulars/C31.pdf.

Barrick, J.E. and Wahlman, G.P. 2019. Conodont and fusulinid biostratigraphy of the Strawn Group (Desmoinesian, Middle Pennsylvanian) and lower part of the" Wolfcamp Shale"(Missourian-Virgilian, Late Pennsylvanian) in the northern Midland Basin, West Texas. Stratigraphy 16:65–86.

Blagoderov, V., Kitching, I.J., Livermore, L., Simonsen, T.J. and Smith, V.S. 2012. No specimen left behind: industrial scale digitization of natural history collections. ZooKeys 133–46. Pensoft Publishers. doi:10.3897/zookeys.209.3178.

Bloice, M.D., Roth, P.M. and Holzinger, A. 2019. Biomedical image augmentation using Augmentor. Bioinformatics. doi:10.1093/bioinformatics/btz259.

Carranza-Rojas, J., Goeau, H., Bonnet, P., Mata-Montero, E. and Joly, A. 2017. Going deeper in the automated identification of Herbarium specimens. BMC Evolutionary Biology 17:181. BioMed Central. doi:10.1186/s12862-017-1014-z.

Chollet, F. and others. 2015. Retrieved from https://keras.io.

Culver, S.J. 2003. Benthic foraminifera across the Cretaceous–Tertiary (K–T) boundary: a review. Marine Micropaleontology 47:177–226. Elsevier. doi:10.1016/S0377-8398(02)00117-2.

Dumoulin, V. and Visin, F. 2016. A guide to convolution arithmetic for deep learning. ArXiv e-prints.

Ellwood, E.R., Dunckel, B.A., Flemons, P., Guralnick, R., Nelson, G., Newman, G., Newman, S., Paul, D., Riccardi, G., Rios, N., Seltmann, K.C. and Mast, A.R. 2015. Accelerating the Digitization of Biodiversity Research Specimens through Online Public Participation. BioScience 65:383–396. Oxford University Press. doi:10.1093/biosci/biv005.

Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M. and Thrun, S. 2017. Dermatologist-level classification of skin cancer with deep neural networks. Nature 542:115–118. Nature Publishing Group. doi:10.1038/nature21056.

Farley, M.B. and Armentrout, J.M. 2000. Fossils in the Oil Patch. Geotimes 14–17. Retrieved from http://www.geotimes.org/oct00/oilpatch.html.

Farley, M.B. and Armentrout, J.M. 2002. Tools, Biostratigraphy becoming lost art in rush to find new exploration. Offshore 94–95. Retrieved from https://www.offshore-mag.com/articles/print/volume-62/issue-2/news/biostratigraphy-becoming-lost-art-in-rush-to-find-new-exploration-tools.html.

Glorot, X. and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks, p. In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics.

Gomez Villa, A., Salazar, A. and Vargas, F. 2017. Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. Ecological Informatics 41:24–32. Elsevier. doi:10.1016/J.ECOINF.2017.07.004.

Gooday, A.J. 1994. The Biology of Deep-Sea Foraminifera: A Review of Some Advances and Their Applications in Paleoceanography. PALAIOS 9:14. SEPM Society for Sedimentary Geology. doi:10.2307/3515075.

Han, J., Morris, S.C., Ou, Q., Shu, D. and Huang, H. 2017. Meiofaunal deuterostomes from the basal Cambrian of Shaanxi (China). Nature 542:228–231. Nature Publishing Group. doi:10.1038/nature21072.

He, K., Zhang, X., Ren, S. and Sun, J. 2016. Identity Mappings in Deep Residual Networks, p. 630–645. In Leibe, B., Matas, J., Sebe, N., and Welling, M. (eds.), Computer Vision -- ECCV 2016. Springer International Publishing, Cham.

Hogancamp, N.J. and Manship, L.L. 2016. Comparison of morphometric techniques and the ability to accurately reconstruct the form and distinguish between species of the Palmatolepis winchelli group-Conodonta, Upper Devonian. MICROPALEONTOLOGY 62:439–451. MICRO PRESS 6530 KISSENA BLVD, FLUSHING, NY 11367 USA.

Huang, G., Liu, Z. and Weinberger, K.Q. 2016. Densely Connected Convolutional Networks. CoRR abs/1608.0. Retrieved from http://arxiv.org/abs/1608.06993.

Kobayashi, F. 2012. Late Paleozoic Foraminifers from Limestone Blocks and Fragments of the Permian Tsunemori Formation and Their Connection to the Akiyoshi Limestone Group, Southwest Japan. Paleontological Research 16:219–243. Palaeontological Society of Japan. doi:10.2517/1342-8144-16.3.219.

Kobayashi, F. and Furutani, H. 2019. Late Early Permian Fusulines along Gongendani, South of Mt. Ryozen, Shiga Prefecture, Central Japan. Paleontological Research 23:131. Palaeontological Society of Japan. doi:10.2517/2018PR014.

Koch, G.R. 2015. Siamese Neural Networks for One-Shot Image Recognition, p. .

Kong, S., Punyasena, S. and Fowlkes, C. 2016. Spatially Aware Dictionary Learning and Coding for Fossil Pollen Identification, p. 1305–1314. 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE. doi:10.1109/CVPRW.2016.165.

Kossovaya, O.L., Novak, M. and Weyer, D. 2016. Large-sized Early Permian “caninioid” corals from the Karavanke Mountains, Slovenia. Journal of Paleontology 90:1049–1067. Paleontological Society. doi:10.1017/jpa.2016.105.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. 2012. ImageNet Classification with Deep Convolutional Neural Networks, p. 1097–1105. Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. NIPS’12. Curran Associates Inc., USA. Retrieved from http://dl.acm.org/citation.cfm?id=2999134.2999257.

Kucera, M. 2007. Chapter Six Planktonic Foraminifera as Tracers of Past Oceanic Environments, p. 1, 213–262. In Hillaire–Marcel, C. and Vernal, A. De (eds.), Proxies in Late Cenozoic Paleoceanography. Developments in Marine Geology. Elsevier. doi:https://doi.org/10.1016/S1572-5480(07)01011-1.

Lake, B.M., Salakhutdinov, R. and Tenenbaum, J.B. 2015. Human-level concept learning through probabilistic program induction. Science (New York, N.Y.) 350:1332–8. American Association for the Advancement of Science. doi:10.1126/science.aab3050.

LeCun, Y., Bengio, Y. and Hinton, G. 2015. Deep learning. Nature 521:436–444. Nature Publishing Group. doi:10.1038/nature14539.

Machine Learning Glossary | Google Developers. . Retrieved January 21, 2019, from https://developers.google.com/machine-learning/glossary/#top_of_page.

Norouzzadeh, M.S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M.S., Packer, C. and Clune, J. 2018. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. Proceedings of the National Academy of Sciences of the United States of America 115:E5716–E5725. National Academy of Sciences. doi:10.1073/pnas.1719367115.

Olah, C., Mordvintsev, A. and Schubert, L. 2017. Feature Visualization. Distill. doi:10.23915/distill.00007.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K. and Mordvintsev, A. 2018. The Building Blocks of Interpretability. Distill. doi:10.23915/distill.00010.

Pan, S.J. and Yang, Q. 2010. A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering 22:1345–1359. doi:10.1109/TKDE.2009.191.

Pires de Lima, R., Bonar, A., Coronado, D.D., Marfurt, K. and Nicholson, C. 2019. Deep convolutional neural networks as a geological image classification tool. The Sedimentary Record 17:4–9. doi:10.210/sedred.2019.2.

Pires de Lima, R., Suriamin, F., Marfurt, K.J. and Pranter, M.J. 2019. Convolutional neural networks as aid in core lithofacies classification. Interpretation 7:SF27–SF40. doi:10.1190/INT-2018-0245.1.

Ranaweera, K., Harrison, A.P., Bains, S. and Joseph, D. 2009. Feasibility of computer-aided identification of foraminiferal tests. Marine Micropaleontology 72:66–75. Elsevier. doi:10.1016/J.MARMICRO.2009.03.005.

Roozpeykar, A. and Moghaddam, I.M. 2016. Benthic foraminifera as biostratigraphical and paleoecological indicators: An example from Oligo-Miocene deposits in the SW of Zagros basin, Iran. Geoscience Frontiers 7:125–140. Elsevier. doi:10.1016/J.GSF.2015.03.005.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C. and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision 115:211–252. Springer US. doi:10.1007/s11263-015-0816-y.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.-C. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. ArXiv e-prints.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D. and Lillicrap, T.P. 2016. One-shot Learning with Memory-Augmented Neural Networks. CoRR abs/1605.0. Retrieved from http://arxiv.org/abs/1605.06065.

Sevillano, V. and Aznarte, J.L. 2018. Improving classification of pollen grain images of the POLEN23E dataset through three different applications of deep learning convolutional neural networks. (S. Rutherford, Ed.)PLOS ONE 13:e0201807. Public Library of Science. doi:10.1371/journal.pone.0201807.

Simonyan, K. and Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv e–prints. Retrieved August 6, 2018, from http://arxiv.org/abs/1409.1556.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15:1929–1958. JMLR.org. Retrieved from http://dl.acm.org/citation.cfm?id=2627435.2670313.

Stevens, C.H. and Stone, P. 2009. New Fusulinids from Lower Permian Turbidites at Conglomerate Mesa, Southeastern Inyo Mountains, East-central California. Journal of Paleontology 83:399–404. doi:10.1666/08-162.1.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. 2014. Going Deeper with Convolutions. CoRR abs/1409.4. Retrieved from http://arxiv.org/abs/1409.4842.

Szegedy, C., Toshev, A. and Erhan, D. 2013. Deep Neural Networks for Object Detection, p. 2553–2561. In Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K.Q. (eds.), Advances in Neural Information Processing Systems 26. Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/5207-deep-neural-networks-for-object-detection.pdf.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. 2015. Rethinking the Inception Architecture for Computer Vision. CoRR abs/1512.0. Retrieved June 27, 2018, from http://arxiv.org/abs/1512.00567.

Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B. and Liang, J. 2016. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? IEEE Transactions on Medical Imaging 35:1299–1312. doi:10.1109/TMI.2016.2535302.

Waddell, D.E. 1966. Pennsylvanian fusulinids in the Ardmore Basin - Love and Carter counties, Oklahoma. Oklahoma Geological Survey Bulletin 113.

Wahlman, G.P. 2019. Pennsylvanian and Lower Permian Fusulinid Biostratigraphy of the Permian Basin Region, Southwestern USA, p. 167–227. In Ruppel, S.C. (ed.), Anatomy of a Paleozoic Basin: The Permian Basin, USA, Volume 1, Bureau of Economic Geology Report of Investigations 285; AAPG Memoir 118.

Williams, T.E. 1963. Fusulinidae of the Hueco Group (Lower Permian) Hueco Mountains, Texas. Peabody Museum, Yale University.

Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. 2014. How transferable are features in deep neural networks? Advances in Neural Information Processing Systems 27:3320–3328. Retrieved August 6, 2018, from http://arxiv.org/abs/1411.1792.

Zhao, Z.-Q., Zheng, P., Xu, S. -t. and Wu, X. 2018. Object Detection with Deep Learning: A Review. ArXiv e-prints.

Zhong, B., Ge, Q., Kanakiya, B., Marchitto, R.M.T. and Lobaton, E. 2017. A comparative study of image classification algorithms for Foraminifera identification, p. 1–8. 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE. doi:10.1109/SSCI.2017.8285164.

Chapter 5: Convolutional neural network for remote sensing scene classification: transfer learning analysis

Rafael Pires de Lima[1,2], Kurt J. Marfurt[1],

[1]School of Geology and Geophysics, The University of Oklahoma, 100 East Boyd Street, RM 710, Norman, Oklahoma, 73019, USA

[2]The Geological Survey of Brazil – CPRM, 55 Rua Costa, São Paulo, São Paulo, Brazil

Preface

Here I show more details of the use of transfer learning making use of remote sensing data. I use different network structures, optimizers, and datasets to evaluate the performance of transfer learning versus training randomly initialized weights. This chapter will be submitted to a remote sensing journal.

Abstract

Remote sensing image scene classification can provide significant value, ranging from forest fire monitoring to land use and land cover classification. Beginning with the first aerial photographs of the early 20th century to the satellite imagery of today, the amount of remote sensing data has increased geometrically with a higher resolution. The need to analyze these modern digital data motivated research to accelerate remote sensing image classification. Fortunately, great advances have been made by the computer vision community to classify natural images or photographs taken with an ordinary camera. Natural image datasets can range up to millions of samples, and are therefore amenable to deep learning techniques. Many fields of science, remote sensing included, were able to exploit the success of natural image classification by convolutional neural network models using a technique commonly called transfer learning. We provide a systematic review of transfer learning application for scene classification using different datasets and different deep learning models. We evaluate how the specialization of convolutional neural network models affect the transfer learning process by splitting original models in different points. As expected, we find the choice of hyperparameters used to train the model have a significant influence on the final performance of the models. Curiously, we find transfer learning from models trained on larger more generic natural images datasets outperformed transfer learning from models trained directly on smaller on remote sensing dataset. Nonetheless, results show that transfer learning provides a powerful tool for remote sensing scene classification.

Keywords

Remote sensing, convolutional neural networks, transfer learning

Glossary

This short glossary provides common denominations in machine learning applications and used throughout the manuscript. Please refer to Google's machine learning glossary for a more detailed list of terms (Google, 2019).

- **Accuracy**: the ratio between the number of correct classifications and the total number of classifications performed. Values range from 0.0 to 1.0 (equivalently, 0% to 100%). A perfect score of 1.0 means all classifications were correct whereas a score of 0.0 means all classifications were incorrect.
- **Convolution**: a mathematical operation that combines input data and a convolutional kernel producing an output. In machine learning applications, a convolutional layer uses the convolutional kernel and the input data to train the convolutional kernel weights.
- **Convolutional Neural Networks (CNN)**: a neuron network architecture in which at least one layer is a convolutional layer.
- **Deep neural networks (DNN)**: an artificial neural network model containing multiple hidden layers.
- **Fine Tuning**: a secondary training step to further adjust the weights of a previously trained model so the model can better achieve a secondary task.
- **Label:** the names applied to an instance, sample, or example (for image classification, an image) associating it with a given class.
- **Layer:** a group of neurons in a machine learning model that processes a set of input features.
- **Machine Learning (ML)**: a model or algorithm that is trained and learns from input data rather than from externally specified parameters.

- **Softmax**: A function that calculates probabilities for each possible class over all different classes. The sum of all probabilities adds to 1.0. The softmax equation $S(\boldsymbol{x_i})$ computed over k classes is given by: $S(\boldsymbol{x_i}) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}$
- **Training**: the iterative process of finding the most appropriate weights of a machine learning model.
- **Transfer Learning**: a technique that uses information learned in a primary machine learning task to perform a secondary machine learning task.
- **Weights**: the coefficients of a machine learning model. In a simple linear equation, the slope and intercept are the weights of the model. In CNNs, the weights are the convolutional kernel values. The training objective is to find the ideal weights of the machine learning model.

1. Introduction

Over the past decades, remote sensing has experienced dramatic changes in the data quality, spatial resolution, shorter revisit times, and area covered available. Emery and Camps (2017) reported that our ability to observe the Earth from low Earth orbit and geostationary satellites have been continuously improving. Such an increase requires a significant change in the way we use and manage remote sensing images. Zhou et al. (2018a) noted that the increased spatial resolution makes it possible to develop novel approaches, providing new opportunities for advancing remote sensing image analysis and understanding, thus allowing us to study the ground surface in greater detail. However, the increase in data available has resulted in important challenges of how to properly manage the imagery collection.

One of the fundamental remote sensing tasks is scene classification. Cheng et al. (2017) defined scene classification as the categorization of remote sensing images into a discrete set of meaningful land cover and land use classes. Scene classification is a fundamental remote sensing task and important for many practical remote sensing applications, such as urban planning (e.g. Xiao and Zhan, 2009), land management (e.g. Skidmore et al., 1997), and to monitoring and or characterizing wild fires monitor or characterize wild fires (e.g. Lentile et al., 2006; Daldegan et al., 2019), among other applications. Such ample use of remote sensing image classification led many researchers to investigate techniques to quickly classify remote sensing data and accelerate image retrieval.

Conventional scene classification techniques rely on low-level visual feature to represent the images of interest. Such low-level features can be global or local. Global features are extracted from the entire remote sensing image, such as color (spectral) features (e.g. Sebai et al., 2015; Bosilj et al., 2016), texture features (e.g. Shao et al., 2014), and shape features (e.g. Scott et al., 2011). Local features, like Scale Invariant Feature Transform (SIFT) (Lowe, 2004) are extracted from image patches that are centered about a point of interest. Zhou et al. (2018a) observed that the remote sensing community makes use of the properties of local features and proposed several methods for remote sensing image analysis. However, these global and local features are hand-crafted. Furthermore, the development of such features is time consuming and oftentimes depend on ad-hoc or heuristic design decisions. For these reasons, the extraction of low-level global and local features are suboptimal for some scene classification tasks. Hu et al. (2015) remarked that the performance of remote sensing scene classification has only slightly improved in recent years. The main reason remote sensing scene classification only marginally improved due to the fact that the approaches relying on low-level features are incapable of generating sufficiently powerful feature representations for remote sensing scenes. Hu et al. (2015) concluded that the more representative and higher-level features, which are abstractions of the lower-level features, are desirable and play a dominant role in scene classification task. The extraction of high-level features promises to be one of the main advantages of deep learning methods. As observed by Yang et al. (2018), one of the reasons for the attractiveness of deep learning models is due the models' capacity to discover effective feature transformations for the desired task.

Recently, the deep learning methods (LeCun et al., 2015) are being applied in many fields of science and industry. Current progress in deep learning models, specifically deep convolutional neural networks (CNN) architectures, have improved the state-of-the-art in visual object recognition and detection, speech recognition and many other fields of study (LeCun et al., 2015). The model described by Krizhevsky et al. (2012), frequently referenced to as AlexNet, is considered a breakthrough and influenced the rapid adoption of DL in the computer vision field (LeCun et al., 2015). CNNs currently are the dominant method in the vast majority image classification, segmentation, and detection tasks due to their remarkable performance in many benchmarks, e.g. the MNIST handwritten database (LeCun, 1998) and the ImageNet dataset (Russakovsky et al., 2015), a large dataset with millions of natural images. In 2012 AlexNet used a five-layer deep CNN model to win the ImageNet Large Scale Visual Recognition Competition. Now, many CNN models use twenty to hundreds of layers. Huang et al. (2016) proposed models with thousands of layers. Due to the vast number of operations performed in deep CNN models, it is often difficult to discuss the interpretability, or the degree to which a decision taken by a model can be interpreted. Thus, CNN interpretability itself remains a research topic (e. g. Simonyan et al., 2013; Olah et al., 2017, 2018; Yin et al., 2017).

Despite CNNs' powerful feature extraction capabilities, Hu et al. (2015) and others found that in practice it is difficult to train CNNs with small datasets. However, Yosinski et al. (2014) and Yin et al. (2017) observed that the parameters learned by the layers in many CNN models trained on images exhibit a very common behavior. The layers closer to the input data tend to learn general features, resulting in convolutional operators akin to edge detection filters, smoothing, or color filters. Then there is a transition to features more specific to the dataset on which the model is trained. These general-specific CNN layer feature transition lead to the development of transfer learning (Caruana, 1995; Bengio, 2012; Yosinski et al., 2014). In transfer learning, the filters learned by a CNN model on a primary task are applied to an unrelated secondary task. The primary CNN model can be used a as feature extractor, or as a starting point for a secondary CNN model.

Even though large datasets help the performance of CNN models, the use of transfer learning facilitated the application of CNN techniques to other scientific fields that have less available data. For example, Carranza-Rojas et al. (2017) used transfer learning for herbarium specimens classification, Esteva et al. (2017) for dermatologist-level classification of skin cancer classification, Pires de Lima et al. (2019d) for oil field drill core images, Duarte-Coronado et al. (2019) for the estimation of porosity in thin section images, Pires de Lima et al. (2019c) and Pires de Lima et al. (2019a) for the classification of a variety of geoscience images. Transfer learning is also widely used in the remote sensing field. For example, Hu et al. (2015) performed an analysis of the use of transfer learning from pretrained CNN models to perform remote sensing scene classification. Chen et al. (2018) used transfer learning for airplane detection, Rostami et al. (2019) for classifying Synthetic Aperture Radar images, Weinstein et al. (2019) for the localization of tree-crowns using Light Detection and Ranging RGB images.

Despite the success of transfer learning in applications in which the secondary task is significantly different from the primary task(e.g. Esteva et al., 2017; Huot et al., 2018; Pires de Lima et al., 2019b), the remark that the effectiveness of transfer learning is expected to decline as the primary and secondary tasks become less similar (Yosinski et al., 2014) is a question commonly raised and still very present in many research fields. Although Yosinski et al. (2014) concluded that using transfer learning from distant tasks perform better than training CNN models from scratch (with randomly initialized weights), it remains unclear how the amount of data or the model used can influence the models' performance.

Here we investigate the performance of transfer learning from CNNs pre-trained on natural images for remote sensing scene classification versus CNNs trained from scratch only on the remote sensing scene classification dataset themselves. We evaluate different depths of two popular CNN models –VGG 19 (Simonyan and Zisserman, 2014), and Inception V3 (Szegedy et al., 2015) – using three different sized remote sensing datasets. Section 2 describes the datasets, section 3 provides details on the methods we apply for analysis. Section 4 shows the results followed by a discussion in section 5. We summarize our findings in section 6.

2. Data

This section provides some details about the datasets we use in our experiments as well as the number of samples for each one of the datasets. We use a 70%-10%-20% split between training, validation, and test sets.

2.1 UCMerced: UC Merced dataset

Introduced by Yang and Newsam (2010), the UC Merced dataset (UCMD) is a land use image dataset containing 21 classes, each class with 100 samples. The images are 256 × 256 pixels, with a spatial resolution of 0.3 m per pixel. The images were manually cropped from the publicly available images USGS National Map Urban Area Imagery collection for various urban areas around the United States. Zhou et al. (2018) observed that the UCMD dataset has many similar or overlapping classes, e.g. sparse residential, medium residential, and dense residential. This similarity combined with the small number of samples per class makes the UCMD a challenging dataset for machine learning classification. Table 8 shows the data split between training, validation, and test sets, as well as the total number of samples for all classes in the UCMD dataset. The dataset is available for download at http://weegee.vision.ucmerced.edu/datasets/landuse.html

Table 8: number of samples for training, validation, and test used for the UCMD dataset.

Class	*Training*	*Validation*	*Test*	*Total*
Agricultural	70	10	20	100
Airplane	70	10	20	100
Baseball diamond	70	10	20	100
Beach	70	10	20	100
Buildings	70	10	20	100
Chaparral	70	10	20	100
Dense residential	70	10	20	100
Forest	70	10	20	100
Freeway	70	10	20	100
Golf course	70	10	20	100
Harbor	70	10	20	100
Intersection	70	10	20	100
Medium residential	70	10	20	100
Mobile home park	70	10	20	100
Overpass	70	10	20	100
Parking lot	70	10	20	100
River	70	10	20	100
Runway	70	10	20	100
Sparse residential	70	10	20	100
Storage tanks	70	10	20	100
Tennis court	70	10	20	100

2.2 AID: Aerial Image Dataset

Xia et al. (2017) presented the Aerial Image Dataset (AID), a remote sensing dataset with 10,000 images. The dataset comprises 30 classes, the number of samples of each range from 220 to 420. The images are 600 x 600 pixels, with a spatial resolution varying from 0.5 to 8 m per pixel. The images in AID were extracted from Google Earth imagery, coming from different remote imaging sensors. Unlike the UCMD, the images from AID are chosen from different countries and regions around the world, mainly in China, the United States, England, France, Italy, Japan, and Germany. Table 9 shows the data split between training, validation, and test sets, as well as the total number of samples for all classes in the AID dataset. The dataset is available for download at http://captain.whu.edu.cn/WUDA-RSImg/aid.html.

Table 9: number of samples for training, validation, and test used for the AID dataset.

Class	*Training*	*Validation*	*Test*	*Total*
Airport	252	36	72	360
Bare land	217	31	62	310
Baseball field	154	22	44	220
Beach	280	40	80	400
Bridge	252	36	72	360
Center	182	26	52	260
Church	168	24	48	240
Commercial	245	35	70	350
Dense residential	287	41	82	410
Desert	210	30	60	300
Farmland	259	37	74	370
Forest	175	25	50	250
Industrial	273	39	78	390
Meadow	196	28	56	280
Medium residential	203	29	58	290
Mountain	238	34	68	340
Park	245	35	70	350
Parking	273	39	78	390
Playground	259	37	74	370
Pond	294	42	84	420
Port	266	38	76	380
Railway station	182	26	52	260
Resort	203	29	58	290
River	287	41	82	410
School	210	30	60	300
Sparse residential	210	30	60	300
Square	231	33	66	330
Stadium	203	29	58	290
Storage tanks	252	36	72	360
Viaduct	294	42	84	420

2.3 PatternNet

Described by Zhou et al. (2018), PatternNet is a large-scale high-resolution remote sensing dataset. PatternNet contains 38 classes, each class with 800 samples. The images are 256 x 256 pixels, with a spatial resolution varying from 0.062 to 4.7 m per pixel. The PatternNet images were collected from Google Earth imagery or via the Google Map API for US cities. Table 10 shows the data split between training, validation, and test sets, as well as the total number of samples for all classes in the PatternNet dataset. The dataset is available for download at https://sites.google.com/view/zhouwx/dataset.

Table 10: number of samples for training, validation, and test used for the PatternNet dataset.

Class	*Training*	*Validation*	*Test*	*Total*
Airplane	560	80	160	800
Baseball field	560	80	160	800
Basketball court	560	80	160	800
Beach	560	80	160	800
Bridge	560	80	160	800
Cemetery	560	80	160	800
Chaparral	560	80	160	800
Christmas tree farm	560	80	160	800
Closed road	560	80	160	800
Coastal mansion	560	80	160	800
Crosswalk	560	80	160	800
Dense residential	560	80	160	800
Ferry terminal	560	80	160	800
Football field	560	80	160	800
Forest	560	80	160	800
Freeway	560	80	160	800
Golf course	560	80	160	800
Harbor	560	80	160	800
Intersection	560	80	160	800
Mobile home park	560	80	160	800
Nursing home	560	80	160	800
Oil gas field	560	80	160	800
Oil well	560	80	160	800
Overpass	560	80	160	800
Parking lot	560	80	160	800
Parking space	560	80	160	800
Railway	560	80	160	800
River	560	80	160	800
Runway	560	80	160	800
Runway marking	560	80	160	800
Shipping yard	560	80	160	800
Solar panel	560	80	160	800
Sparse residential	560	80	160	800
Storage tank	560	80	160	800
Swimming pool	560	80	160	800
Tennis court	560	80	160	800
Transformer station	560	80	160	800
Wastewater treatment plant	560	80	160	800

3. Methods

To better understand the effects of different approaches and techniques used for transfer learning with remote sensing datasets, we perform two major experiments using the models presented in section 3.1. The first experiment in section 3.2 compares different optimization methods. The second experiment in section 3.3 aims to investigate the sensitivity of transfer learning to the level of specialization of the original trained CNN model. The experiment in section 3.2 also compares the results of transfer learning and training a model with randomly initialized weights.

The choice of hyperparameters can have a strong influence in CNN performance. Nonetheless, our main objective here is to investigate transfer learning results rather than maximize performance. Therefore, unless otherwise noted, we maintain the same hyperparameters specified in Table 11 for all training in all experiments. The models are trained using Keras (Chollet and others, 2015), with TensorFlow as its backend (Abadi et al., 2016). When kernels are initialized, we use the Glorot uniform (Glorot and Bengio, 2010) distribution of weights.

Table 11: Training hyperparameters

Optimizer	**Stochastic gradient descent**
Kernel initializer	Glorot uniform
Batch size	32
Epochs	100
Loss function	Cross entropy

3.1 Model split

To evaluate the transfer learning process from natural images to remote sensing datasets, we use VGG19 and Inception V3 models and train a small classification network on top of such models. We refer to the original CNN model structure, part of VGG19 or part of Inception V3, as the "base model", and the small classification network as the "top model" (Figure 1). The top model is composed of an average pooling, followed by one fully connected layer with 512 neurons, a dropout layer (Srivastava et al., 2014) used during training, and a final fully connected layer with a softmax output where the number of neurons is dependent on the number of classes for the task (i.e., 21 for UCMerced, 30 for AID, 38 for PatternNet). The dropout is a simple technique useful to avoid overfitting in which random connections are disabled during training. Note the top model will be specific to the secondary task and for each one of the datasets, whereas the base model, when containing the weights learned during training for the primary task, will have its layers presenting the transition from general to specific features. The models we used were primarily trained on the ImageNet dataset and are available online (e.g. through Keras or TensorFlow websites). We evaluate how dependent the transfer learning process is on the transition from general to specific features by extracting features in three different positions for each one of the retrained models and we denominate them "shallow", "intermediate", and "deep" (Figure 2). The shallow experiment uses the initial blocks of the base models to extract features and adds the top model. The intermediate experiment extracts the block somewhere in the middle of the base model. Finally, the deep experiment uses all the blocks of the original base model, except the original final classification layers.

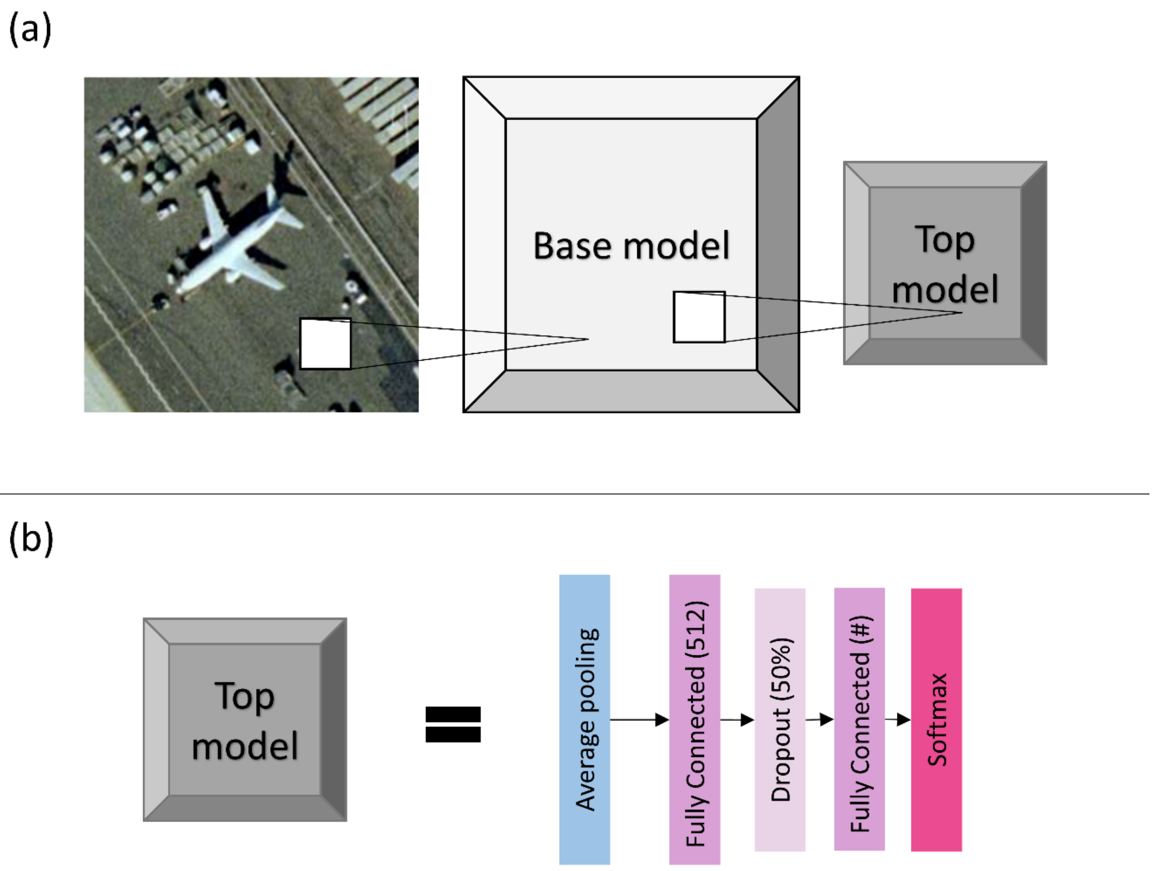


Figure 29: Visualization of the models used. (a) shows a sample image from UCMerced, the base model, and the top model. (b) provides more details for the top model. The base model is dependent on the CNN architecture used for transfer learning and it is detailed in Figure 10. Top model is the same for all experiments. Note the pound sign "#" represents the number of classes, which depends on the dataset used.

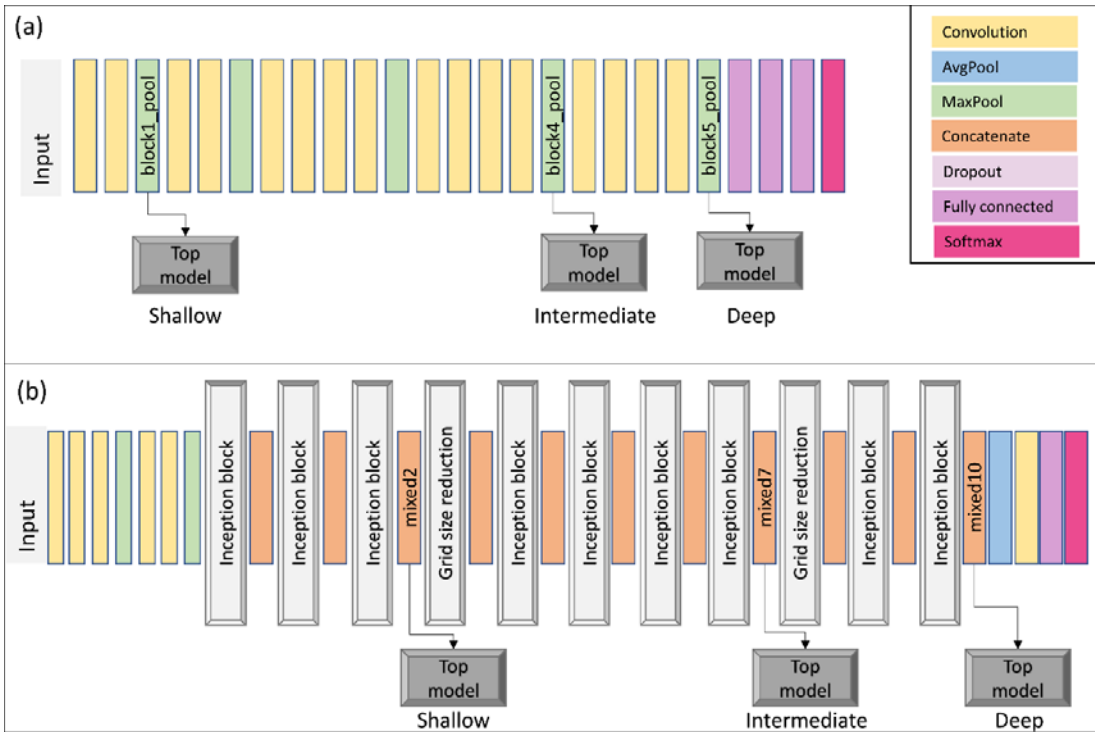


Figure 30: Visual representation of the models used. In both panels, data flows from left to right. Both panes use the same color code for layer representation. (a) shows the VGG19 shallow, intermediate, and deep models – based on the naming convention we are using. (b) shows the Inception V3 shallow, intermediate, and deep models. For easier reference, we wrote the layer names (as implemented in Keras) for each one of the layers we used to split the original CNN models. Note for each one of the depth levels (shallow, intermediate, deep), we simple use the model up to the detour and connect it with our top model (e.g., when training VGG19 shallow, the data goes through two convolutional layers, one max pooling layers, and exits into our top model). Please refer to Simonyan and Zisserman (2014) and Szegedy et al. (2015) for details on VGG19 and Inception V3 respectively.

3.2 Stochastic gradient descent vs adaptive optimization methods

In the search for the global minima, optimization algorithms frequently use the gradient descent strategy. To compute the gradient of the loss function, we sum the error of each sample. Using our PatternNet data split as example, we first loop through all training set containing 21,280 samples before updating the gradient. Therefore, to move a single step towards the minima, we compute the error 21,280 times. A common approach to avoid computing the error for all training samples before moving a step is to use stochastic gradient descent (SGD).

The SGD uses a straightforward approach; instead of using the sum of all training errors (the loss), SGD uses the error gradient of a single sample at each iteration. Bottou (2010) observes that SGD show good performance for large-scale problems. SGD is the building block used by many optimization algorithms that apply some variation to achieve better convergence rates (e.g. Duchi et al., 2011; Tieleman and Hinton, 2012). Kingma and Ba (2014) note that SGD has a great practical importance in many fields of science and engineering and propose Adam, a method for efficient stochastic optimization. Ruder (2016) recommends using Adam as the best overall optimization choice.

However, Wilson et al. (2017) reported that the solutions found by adaptive methods (such as Adam) have a worse generalization than SGD, even though solutions found by adaptive optimization methods have a better performance on the training set. Our optimization experiment is straightforward: we compare the training, validation losses and the test accuracy for the UCMerced dataset using different optimization methods: SGD, Adam, and Adamax – a variant of Adam that makes use of the infinity norm, also described in Kingma and Ba (2014). We perform such analysis using the shallow-intermediate-deep VGG19 and shallow-intermediate-deep Inception V3 to fit the UCMerced dataset starting the models with randomly initialized weights.

3.3 General to specific layer transition of CNN models

As mentioned before, many CNN models trained on natural images show a very common characteristic. Layers closer to the input data tend to learn general features, then there is a transition to more specific dataset features. For example, a CNN trained to classify the 21 UCMerced dataset has in its final layer 21 softmax outputs, with each output specifically identifying one of the 21 classes. Therefore, the final layer in this example is very specific for the UCMerced task; the final layer receives a set of features coming from the previous layers and outputs a set of probabilities accounting for the 21 UCMerced classes. These are intuitive notions of general vs specific features that are sufficient for the experiments to be performed. Yosinski et al. (2014) provide a rigorous definition of general and specific features.

To observe how the transition from general to specific features can affect the transfer learning process of remote sensing datasets, we use the shallow, intermediate, and deep VGG19 and Inception V3 described in section3.1. Three training modes are performed: feature extraction, fine tuning, and randomly initialized weights. Feature extraction “locks” (or “freezes”) the pre-trained layers extracted from the base models. Fine tuning starts as feature extraction, with the base model frozen, but eventually allows all the layers of the model to learn. The randomly initialized weights mode starts the entire model with randomly initialized weights after which all the weights are updated during training. Randomly initialized weights is the ordinary CNN training, not a transfer learning process. For the sake of standardization, all modes train the model for 100 epochs. In fine tuning, the first step (part of the model is frozen) is trained for 50 epochs, and the second step (all layers of the model are free to learn) for another 50 epochs.

4. Results

4.1 Stochastic gradient descent vs adaptive optimization methods

We train the shallow, intermediate, and deep VGG19 and Inception V3 models using the UCMerced dataset with different optimizers. Table 5 shows the naming convention we use here.

Figure 3 shows the accuracy per epoch for each one of the trained models, with each one of the optimizers. Figure 4 shows the accuracy on the test set obtained by each one of the models, with each one of the optimizers. Figure 5 shows the difference in accuracy between the training set and the test set. Table 6 shows a summary of optimizer performance on the test set with the computation of a simple average and median of the accuracy across all tests performed. This test was run using a batch size of 16 on a NVIDIA Quadro M2000.

Table 12: Naming convention and optimizer details

Name	**Optimizer details**
SGD (1e-2)	SGD optimizer with learning rate of 0.01
SGD (1e-2) momentum 0.9	SGD optimizer with learning rate of 0.01 and momentum 0.9
SGD (1e-3)	SGD optimizer with learning rate of 0.001
SGD (1e-3) momentum 0.9	SGD optimizer with learning rate of 0.001 and momentum 0.9
Adam (1e-2)	Adam optimizer with learning rate of 0.01 and default parameters as described in Kingma and Ba (2014)
Adamax (2e-3)	Adamax optimizer with learning rate of 0.01 and default parameters as described in Kingma and Ba (2014)

Table 13: Optimizer performance summary.

Optimizer	**Average accuracy**	**Median Accuracy**
SGD (1e-2)	**0.82**	0.80
SGD (1e-2) momentum 0.9	0.53	0.66
SGD (1e-3)	0.74	0.75
SGD (1e-3) momentum 0.9	0.81	0.82
Adam (1e-2)	0.59	0.81
Adamax (2e-3)	0.59	**0.85**

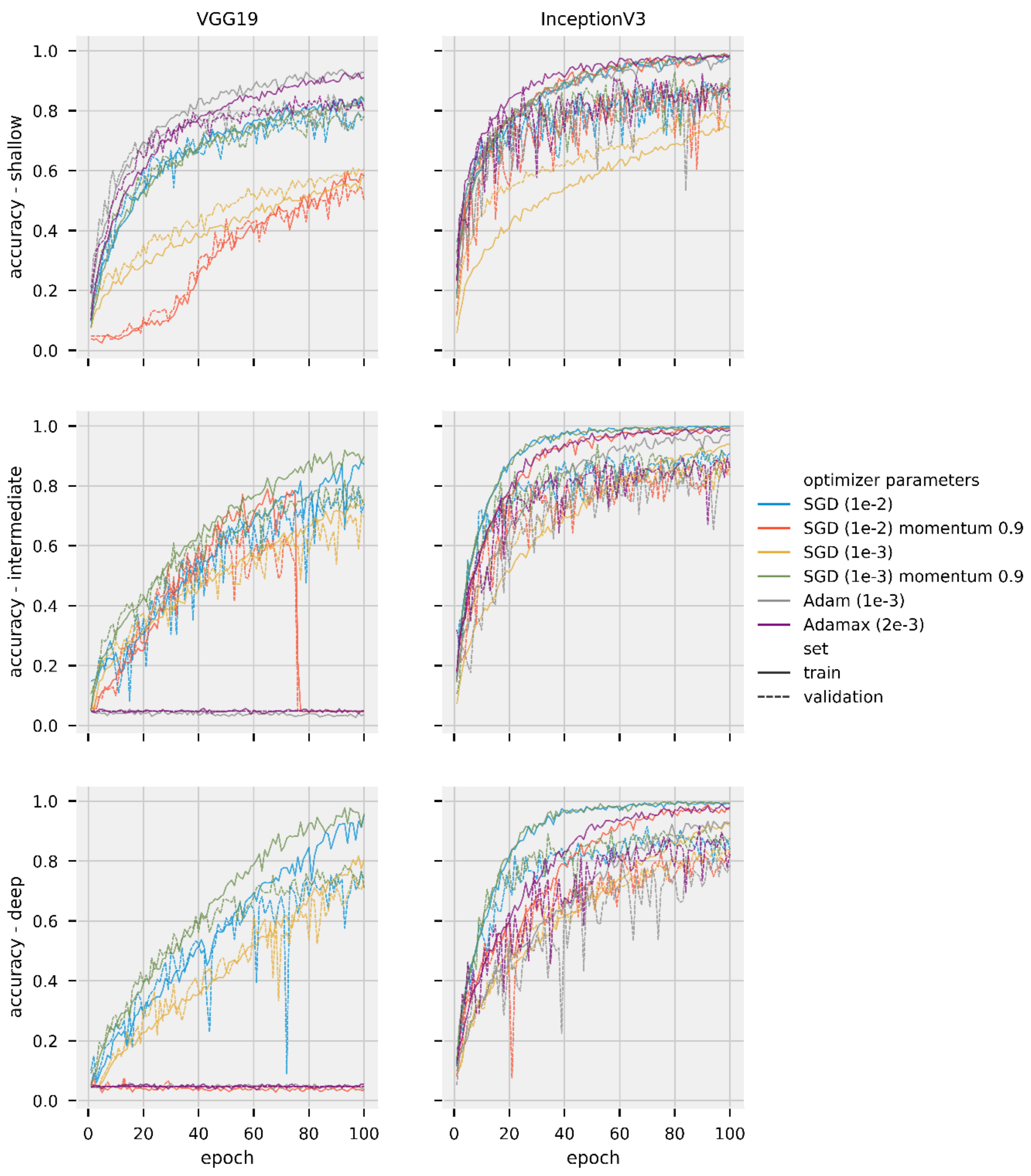


Figure 31: Accuracy per epoch for training and validation sets for different models and optimizers trained on the UCMerced dataset. The left column shows results for VGG19 models. The right column shows results for Inception V3 models. The first row shows shallow models, center shows intermediate, bottom shows deep models. Different colors represent different optimizers. Different and line style represent different datasets (solid for training, dashed for validation).

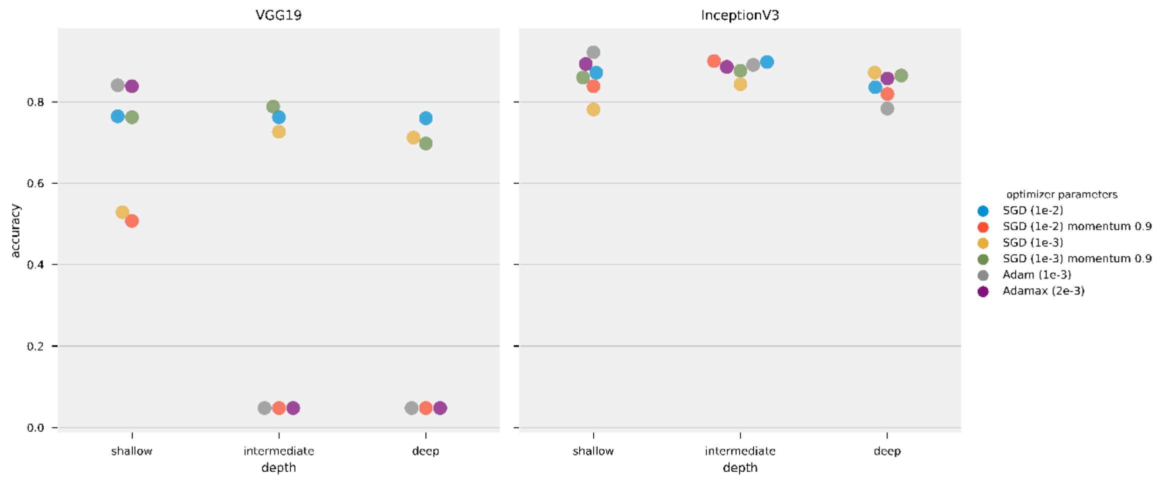


Figure 32: Test set accuracy obtained by the models using different optimizers training on the same UCMerced dataset. The left panel shows VGG 19 results, right panel shows Inception V3 results.

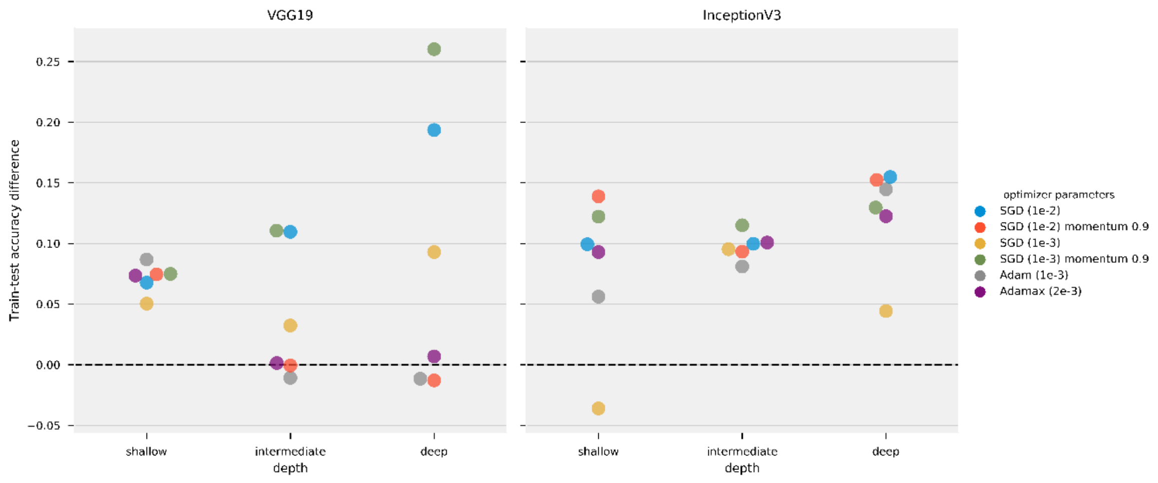


Figure 33: Difference between training set and test set accuracy obtained by the models using different optimizers training on the same UCMerced dataset. The left panel shows VGG 19 results, right panel shows Inception V3 results. Note, as shown in Figure 4, that SGD (1e-2), Adam (1e-2), and Adamax (2e-3) results of theVGG19 intermediate and deep models remained stuck on local minima.

4.2 General to specific layer transition of CNN models

This section shows the results of transfer learning, both feature extraction and fine tuning modes, as well as training the models with randomly initialized weights. Table 7 shows a summary of the best performance Inception V3 and VGG19 trained using SGD (1e-3) momentum 0.9. We chose SGD (1e-3) momentum 0.9 as it is the optimizer with the second-best median in Table 6, and did not become stuck in local minima. The table shows, for each dataset and for each model, which depth and training mode achieved the highest accuracy in the test set. We select AID trained on Inception V3 intermediate, one out of the 54 experiments (three datasets, two models, three depths, three training modes), to provide more details of the training loss-accuracy and the confusion matrix computed for the test set. Figure 5 shows the training and validation loss and accuracy through the training epochs. Figure 6 shows the correspondent confusion matrix computed on the AID test set.

Figure 8 shows an overview of the complete experiment on the test set. The figure shows the test set accuracy for all the datasets, for all the models' depths and training mode. This test was run on a NVIDIA GeForce RTX 2060 and it took roughly six days to complete.

We then select all the six models trained on PatternNet with randomly initialized weights and we perform transfer learning, using the same methodology as before. Thus, we first train CNN models on PatternNet and then apply transfer learning to train on AID and UCMerced. Note this is slightly different than the transfer learning performed before, where we split a single model in three different parts. Here we use the model in their original form (shallow, intermediate, or deep). Loss decays and confusion matrix figures, as well as the complete table with all test accuracies are provided in the supplemental materials. Table 8 shows the best performing Inception V3 and VGG19 for each one of the datasets.

We repeat transfer learning tests with Adamax (2e-3), the optimizer with best median performance on Table 6, although falling in local minima in some tests. Table 9 shows a summary of the best performing Inception V3 and VGG19 trained using Adamax (2e-3) for the three datasets used. Figure 9 shows an overview of the complete experiment on the test set. We did not repeat the PatternNet to AID-UCMerced transfer learning using Adamax (2e-3) as results in Table 14 are generally better than results in Table 16.

Table 14: Best test set accuracy for Inception V3 and VGG19 version for each Dataset using SGD (1e-3) momentum 0.9 optimizer.

Dataset	**model**	**depth**	**mode**	**accuracy**
PatternNet	Inception V3	intermediate	fine tune	**0.997**
	VGG19	deep	fine tune	0.995
AID	Inception V3	intermediate	fine tune	**0.950**
	VGG19	deep	fine tune	0.936
UCMerced	Inception V3	intermediate	fine tune	**0.983**
	VGG19	deep	fine tune	0.981

Table 15: Best test set accuracy for Inception V3 and VGG19 version for each Dataset using SGD (1e-3) momentum 0.9 optimizer to perform transfer learning on models initially trained on PatternNet.

Dataset	**model**	**depth**	**mode**	**accuracy**
AID	InceptionV3	deep	fine tune	**0.838**
	VGG19	intermediate	fine tune	0.833
UCMerced	InceptionV3	intermediate	fine tune	**0.948**
	VGG19	deep	fine tune	0.886

Table 16: Best test set accuracy for Inception V3 and VGG19 version for each Dataset using Adamax (2e-3) optimizer.

Dataset	**model**	**depth**	**mode**	**accuracy**
PatternNet	InceptionV3	intermediate	fine tune	**0.993**
	VGG19	deep	feature extraction	0.983
AID	InceptionV3	intermediate	fine tune	**0.941**
	VGG19	deep	feature extraction	0.889
UCMerced	InceptionV3	deep	fine tune	0.910
	VGG19	deep	feature extraction	**0.943**

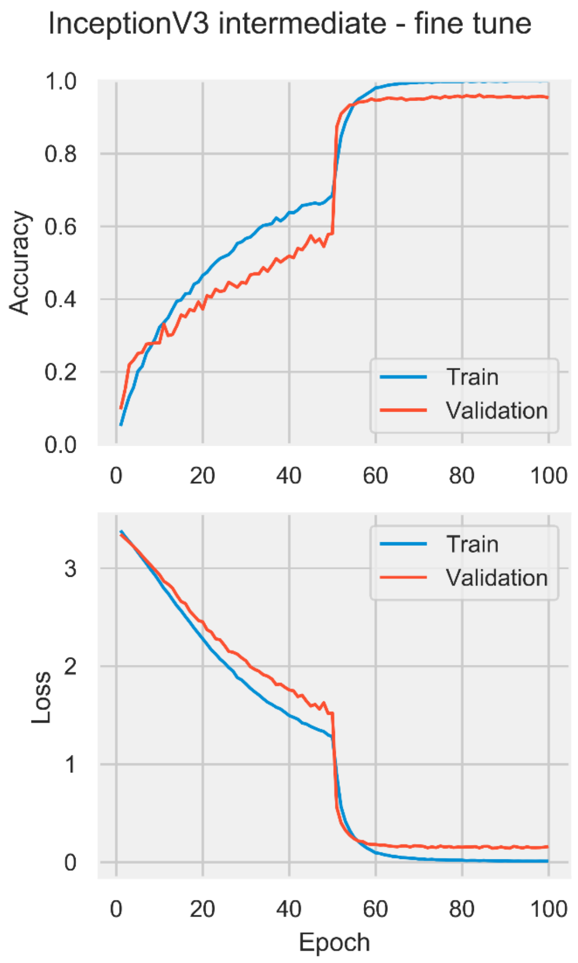


Figure 34: Train and validation loss and accuracy for the Inception V3 intermediate in the fine tune mode trained on the AID dataset using SGD (1e-3) momentum 0.9.

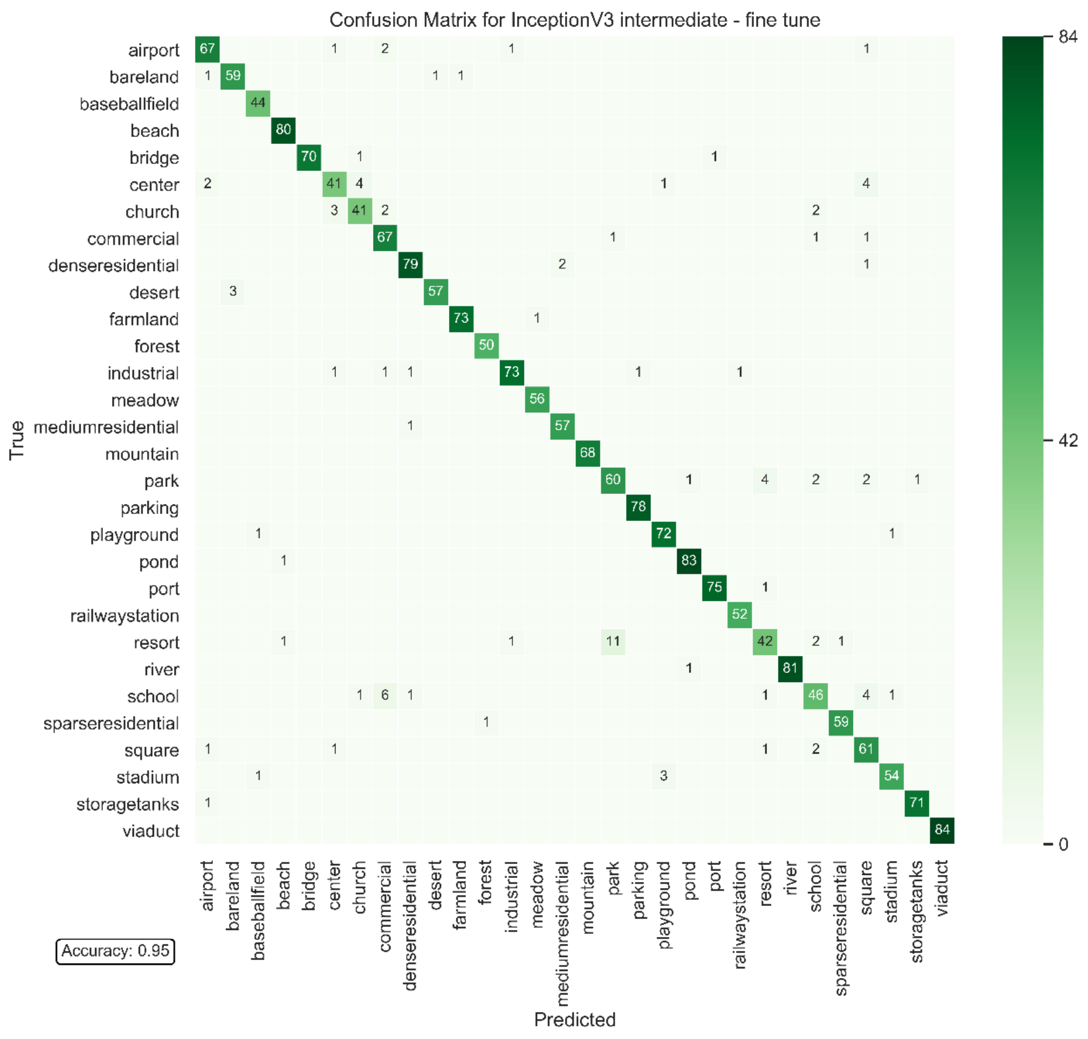


Figure 35: Confusion matrix for the test set of AID dataset for the Inception V3 intermediate in the fine tune mode using SGD (1e-3) momentum 0.9.

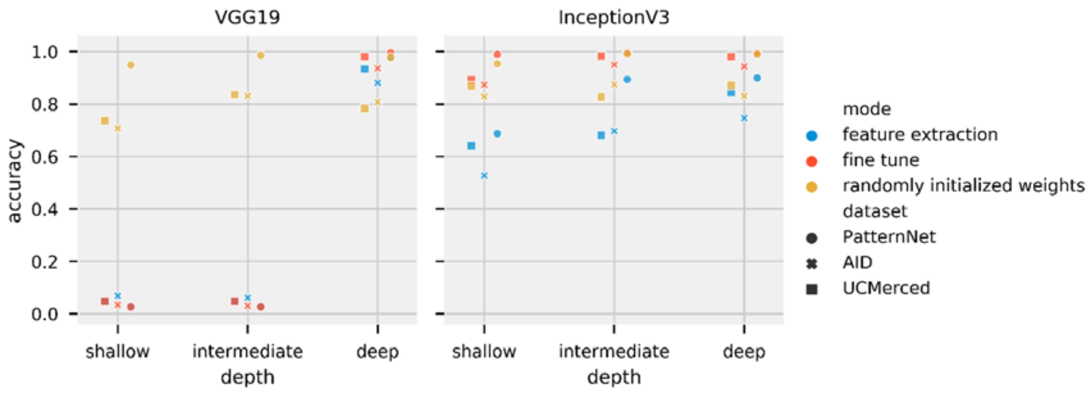


Figure 36: Test set accuracy for all VGG19 and Inception V3 versions trained with all datasets using SGD (1e-3) momentum 0.9. Left panel shows VGG 19 results, right panel shows Inception V3 results. Note VGG19 shallow and intermediate feature extraction and fine tune versions were trapped in local minima.

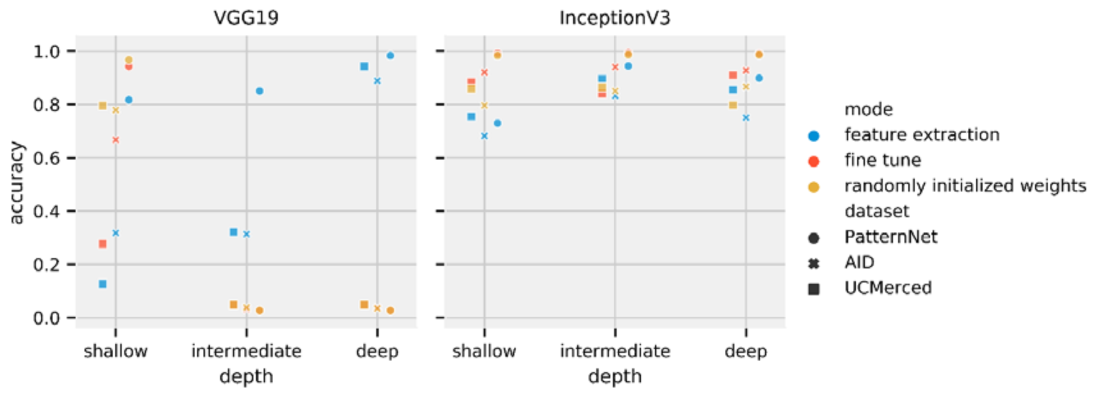


Figure 37: Test set accuracy for all VGG19 and Inception V3 versions trained with all datasets using Adamax (2e-3). Left panel shows VGG 19 results, right panel shows Inception V3 results. Note VGG19 shallow and intermediate feature extraction and fine tuning versions were trapped in local minima.

5. Discussion

Unlike the poor generalization performance of adaptive methods compared to SGD optimizers reported by Wilson et al. (2017) , our results do not find significant differences in performance for the optimizers tested. In fact, results in Figure 5 indicate that for our task, SGDs had a slightly worse performance. SGD (1e-3) momentum 0.9 and SGD (1e-2) had the larger difference between accuracy in training and test set for VGG19 intermediate and deep models. SGD (1e-2) momentum 0.9 had the worst performance for Inception V3 shallow. However, Table 7 showing a summary of SGD (1e-3) momentum 0.9 results are slightly better than the Adamax (2e-3) summary in Table 9. Figure 5 also presents some cases in which the accuracy in the test set was higher than the in the training set, e.g. SGD (1e-3) for Inception V3 shallow model. Validation and test set metrics better than training set metrics can be caused by the dropout layer, as during training less information is available for the model, or simply because of the data split; the training set is generally larger than validation and test sets and can incorporate a higher complexity in its samples.

Even though we selected SGD (1e-3) momentum 0.9 as the optimizer for the transfer learning experiments due to the its performance on the optimizer tests, some models still fell into local minima. This failure shows how the choice of hyperparameters can strongly affect the performance of deep learning models. Neither training from random initial weights (Figure 4) nor transfer learning techniques (Figure 8) are exempt from the possibility to present a poor performance if suboptimal hyperparameters are used. More than a marginal increase in performance, the results show that the models can completely fail when used with inappropriate hyperparameters, even if the task is appropriate for the model.

Using ImageNet data, Yosinski et al. (2014) found that transfer learning, even when applied to a secondary task not similar to the primary task, perform better than training CNN models with randomly initialized weights. Using medical image data, Tajbakhsh et al. (2016) found that fine tuning achieved results comparable to or better than results from training a CNN model with randomly initialized weights. Our results align with their findings. Both Figure 8 and Table 7 show the fine tuning mode of training outperforming randomly initialized weights when using

SGD (1e-3) momentum 0.9. Results in Figure 9 and Table 9 show that transfer learning is the best performing with the Adamax (2e-3) optimizer. However, it seems that the step size (2e-3) is too large for fine tuning in the VGG19 model, such that the VGG19 intermediate and deep models trained on fine tune and randomly initialized weights modes fall in local minima. The primary task (ImageNet, composed of natural images) is not very similar to the secondary task (remote sensing scene classification). While there is a similarity in primary and secondary tasks datasets, such as the number of channels (red-green-blue components), images are from the visible spectra, and some objects might be present in both tasks (e.g. airplanes), the tasks are fundamentally different. Figure 5, however, shows how feature extraction can be limited by the difference in tasks. When the initial layers are frozen, the model cannot properly learn and the model starts to overfit. With the layers unfrozen, the overfitting reduces and accuracy increase. We observed a similar behavior for most of fine tuning tests (all of the loss and accuracy per epoch can be accessed in the supplemental material). Despite feature extraction limitations, the results show that transfer learning is an effective deep learning approach that should not be discarded if the secondary task is not similar to the primary task. In fact, Table 8 presents striking results. Fine tuned models initially trained on PatternNet underperformed fine tuned models trained on ImageNet. Perhaps the first explanation for such underperformance would be that the models are overfitting the PatternNet dataset. However, PatternNet models performed well on the PatternNet test set, which indicates they are not overfitting the training data. We hypothesize that the weaker performance is due to the complexity of the datasets. PatternNet is a dataset created with the objective to provide researchers with clear examples of different remote sensing scenes, whereas the ImageNet is a complex dataset where the intra class variance, i.e. how a single class contains very different samples, is very high. As observed by Cheng et al.

(2017) many remote sensing scene classification datasets have a lack of intraclass sample variations and diversity. These limitations severely limit the development of new approaches especially deep learning-based methods. Thus, CNN models trained on the ImageNet need to develop more generic, perhaps robust, filters to be able to properly identify ImageNet's classes.

6. Conclusions

Our objective with this paper was to investigate the use of transfer learning in the analysis of remote sensing data, as well as how the CNN performance depends on the depth of the network and on the amount of training data available. Our experiments, based on three distinct remote sensing datasets and two popular CNN models, show that transfer learning, specifically fine tuning CNNs is a powerful tool for remote sensing scene classification. Much like the findings in other experiments, the results we found show that transfer from natural images (ImageNet) to remote sensing imagery is possible. Despite the relatively large difference between primary and secondary tasks, transfer learning training mode generally outperformed training a CNN with randomly initialized weights and achieved the best results overall. Curiously, fine tuning models primarily trained on the generic ImageNet dataset overperformed fine tuning models primarily trained on PatternNet dataset. As expected, our results also indicate that for a particular application, the amount of training data available plays a significant role on the performance of the CNN models. We generally observed a larger accuracy difference between transfer learning and training with randomly initialized weights using the smaller UCMerced dataset, whereas accuracy differences were smaller when using the larger PatternNet dataset. Model robustness is also clear on the results. On several instances the VGG19 ended up stuck on local minima, both during optimization testing and during transfer learning testing. The

VGG19 shallow and intermediate models' results exhibit a performance degradation caused by splitting the primary trained CNN model between co-adapted neurons on neighboring layers. VGG19 shallow and intermediate on randomly initialized mode, however, performed satisfactorily. In spite of our simplistic model split without detailed attention to co-adaption of neurons between layers, Inception V3 passed all experiments without falling into local minima.

The results seem to corroborate that feature extraction or fine tuning well-established CNN models offer a practical way to achieve the best performance for remote sensing scene classification. Although fine tune the originally more complex deep models might present satisfactory results, splitting the original model can perhaps improve performance. Note fine tuning Inception V3 intermediate model outperformed Inception V3 deep model. With datasets large enough, randomly initialized weights are also an appropriate choice for training. However, it is often hard to know when a dataset is large enough. Our recommendation is to start from the deep models and try to reduce model's size as it is easier to overfit models with too many weights.

7. Data and materials availability

Data associated with this research are available online. The UC Merced dataset The dataset is available for download at http://weegee.vision.ucmerced.edu/datasets/landuse.html. AID is available for download at for download at http://captain.whu.edu.cn/WUDA-RSImg/aid.html. PatternNet dataset is available for download at https://sites.google.com/view/zhouwx/dataset. The Python scripts used for the analysis of the datasets, as well as the generation of most images, and the supplemental figures are available at https://github.com/raplima/remote_sensing-transfer_learning.

8. Acknowledgments

Rafael acknowledges CNPq (grant 203589/2014-9) for the financial support and CPRM for granting the leave of absence allowing the pursuit of his Ph.D. studies. The funding for the computers used in this project was provided by the industry sponsors of the OU Attribute-Assisted Seismic Processing and Interpretation Consortium.

References

Abadi, M. et al., 2016, TensorFlow: A system for large-scale machine learning, in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16): 265–283.

Bosilj, P., E. Aptoula, S. Lefèvre, and E. Kijak, 2016, Retrieval of Remote Sensing Images with Pattern Spectra Descriptors: ISPRS International Journal of Geo-Information, 5, 228, doi:10.3390/ijgi5120228.

Bottou, L., 2010, Large-Scale Machine Learning with Stochastic Gradient Descent, in Y. Lechevallier, and G. Saporta, eds., Proceedings of COMPSTAT'2010: Physica-Verlag HD, 177–186, doi:.org/10.1007/978-3-7908-2604-3_16.

Carranza-Rojas, J., H. Goeau, P. Bonnet, E. Mata-Montero, and A. Joly, 2017, Going deeper in the automated identification of Herbarium specimens: BMC Evolutionary Biology, 17, 181, doi:10.1186/s12862-017-1014-z.

Chen, Z., T. Zhang, C. Ouyang, Z. Chen, T. Zhang, and C. Ouyang, 2018, End-to-End Airplane Detection Using Transfer Learning in Remote Sensing Images: Remote Sensing, 10, 139, doi:10.3390/rs10010139.

Cheng, G., J. Han, and X. Lu, 2017, Remote Sensing Image Scene Classification: Benchmark and State of the Art: Proceedings of the IEEE, 105, 1865–1883, doi:10.1109/JPROC.2017.2675998.

Chollet, F., and others, 2015, Keras.

Daldegan, G. A., D. A. Roberts, and F. de F. Ribeiro, 2019, Spectral mixture analysis in Google Earth Engine to model and delineate fire scars over a large extent and a long time-series in a rainforest-savanna transition zone: Remote Sensing of Environment, 232, 111340, doi:10.1016/J.RSE.2019.111340.

Duarte-Coronado, D., J. Tellez-Rodriguez, R. Pires de Lima, K. Marfurt, and R. Slatt, 2019, Deep convolutional neural networks as an estimator of porosity in thin-section images for unconventional reservoirs, in SEG Technical Program Expanded Abstracts 2019: Society of Exploration Geophysicists, 3181–3184, doi:10.1190/segam2019-3216898.1.

Duchi, J., E. Hazan, and Y. Singer, 2011, Adaptive Subgradient Methods for Online Learning and Stochastic Optimization: Journal of Machine Learning Research, 12, 2121–2159.

Emery, W., and A. Camps, 2017, Chapter 1 - The History of Satellite Remote Sensing, in W. Emery, and A. Camps, eds., Introduction to Satellite Remote Sensing: Elsevier, 1–42, doi:https://doi.org/10.1016/B978-0-12-809254-5.00001-4.

Esteva, A., B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, 2017, Dermatologist-level classification of skin cancer with deep neural networks: Nature, 542, 115–118, doi:10.1038/nature21056.

Glorot, X., and Y. Bengio, 2010, Understanding the difficulty of training deep feedforward neural networks, in In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics.

Hu, F., G.-S. Xia, J. Hu, and L. Zhang, 2015, Transferring Deep Convolutional Neural Networks for the Scene Classification of High-Resolution Remote Sensing Imagery: Remote Sensing, 7, 14680–14707, doi:10.3390/rs71114680.

Huang, G., Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, 2016, Deep Networks with Stochastic Depth.

Huot, F., B. Biondi, and G. Beroza, 2018, Jump-starting neural network training for seismic problems, in SEG Technical Program Expanded Abstracts 2018: Society of Exploration Geophysicists, 2191–2195, doi:10.1190/segam2018-2998567.1.

Kingma, D. P., and J. Ba, 2014, Adam: A Method for Stochastic Optimization: arXiv e-prints, arXiv:1412.6980.

Krizhevsky, A., I. Sutskever, and G. E. Hinton, 2012, ImageNet Classification with Deep Convolutional Neural Networks, in Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1: Curran Associates Inc., NIPS'12, 1097–1105.

LeCun, Y., 1998, The MNIST database of handwritten digits: http://yann. lecun. com/exdb/mnist/.

LeCun, Y., Y. Bengio, and G. Hinton, 2015, Deep learning: Nature, 521, 436–444, doi:10.1038/nature14539.

Lentile, L. B., Z. A. Holden, A. M. S. Smith, M. J. Falkowski, A. T. Hudak, P. Morgan, S. A. Lewis, P. E. Gessler, and N. C. Benson, 2006, Remote sensing techniques to assess active fire characteristics and post-fire effects: International Journal of Wildland Fire, 15, 319–345.

Lowe, D. G., 2004, Distinctive Image Features from Scale-Invariant Keypoints: International Journal of Computer Vision, 60, 91–110, doi:10.1023/B:VISI.0000029664.99615.94.

Machine Learning Glossary | Google Developers, 2019: <https://developers.google.com/machine-learning/glossary/> (accessed July 21, 2019).

Olah, C., A. Mordvintsev, and L. Schubert, 2017, Feature Visualization: Distill, doi:10.23915/distill.00007.

Olah, C., A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, 2018, The Building Blocks of Interpretability: Distill, doi:10.23915/distill.00010.

Pires de Lima, R., A. Bonar, D. D. Coronado, K. Marfurt, and C. Nicholson, 2019, Deep convolutional neural networks as a geological image classification tool: The Sedimentary Record, 17, 4–9, doi:10.210/sedred.2019.2.

Pires de Lima, R., Y. Lin, and K. J. Marfurt, 2019, Transforming seismic data into pseudo-RGB images to predict CO2 leakage using pre-learned convolutional neural networks weights, in SEG Technical Program Expanded Abstracts 2019: Society of Exploration Geophysicists, 2368–2372, doi:10.1190/segam2019-3215401.1.

Pires de Lima, R., K. Marfurt, D. Duarte, and A. Bonar, 2019, Progress and Challenges in Deep Learning Analysis of Geoscience Images, in 81st EAGE Conference and Exhibition 2019: EAGE, doi:10.3997/2214-4609.201901607.

Pires de Lima, R., F. Suriamin, K. J. Marfurt, and M. J. Pranter, 2019, Convolutional neural networks as aid in core lithofacies classification: Interpretation, 7, SF27–SF40, doi:10.1190/INT-2018-0245.1.

Rostami, M., S. Kolouri, E. Eaton, and K. Kim, 2019, Deep Transfer Learning for Few-Shot SAR Image Classification: Remote Sensing, 11, 1374, doi:10.3390/rs11111374.

Ruder, S., 2016, An overview of gradient descent optimization algorithms: CoRR, abs/1609.0.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, 2015, ImageNet Large Scale Visual Recognition Challenge: International Journal of Computer Vision, 115, 211–252, doi:10.1007/s11263-015-0816-y.

Scott, G. J., M. N. Klaric, C. H. Davis, and C.-R. Shyu, 2011, Entropy-Balanced Bitmap Tree for Shape-Based Object Retrieval From Large-Scale Satellite Imagery Databases: IEEE Transactions on Geoscience and Remote Sensing, 49, 1603–1616, doi:10.1109/TGRS.2010.2088404.

Sebai, H., A. Kourgli, and A. Serir, 2015, Dual-tree complex wavelet transform applied on color descriptors for remote-sensed images retrieval: Journal of Applied Remote Sensing, 9, 095994, doi:10.1117/1.JRS.9.095994.

Shao, Z., W. Zhou, L. Zhang, and J. Hou, 2014, Improved color texture descriptors for remote sensing image retrieval: Journal of Applied Remote Sensing, 8, 083584, doi:10.1117/1.JRS.8.083584.

Simonyan, K., A. Vedaldi, and A. Zisserman, 2013, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps: CoRR, abs/1312.6.

Simonyan, K., and A. Zisserman, 2014, Very Deep Convolutional Networks for Large-Scale Image Recognition: ArXiv e–prints.

Skidmore, A. K., W. Bijker, K. Schmidt, and L. Kumar, 1997, Use of remote sensing and GIS for sustainable land management: ITC journal, 3, 302–315.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 2014, Dropout: A Simple Way to Prevent Neural Networks from Overfitting: Journal of Machine Learning Research, 15, 1929–1958.

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, 2015, Rethinking the Inception Architecture for Computer Vision: CoRR, abs/1512.0.

Tajbakhsh, N., J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, 2016, Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? IEEE Transactions on Medical Imaging, 35, 1299–1312, doi:10.1109/TMI.2016.2535302.

Tieleman, T., and G. Hinton, 2012, Lecture 6.5---RmsProp: Divide the gradient by a running average of its recent magnitude.

Weinstein, B. G., S. Marconi, S. Bohlman, A. Zare, and E. White, 2019, Individual Tree-Crown Detection in RGB Imagery Using Semi-Supervised Deep Learning Neural Networks: Remote Sensing, 11, 1309, doi:10.3390/rs11111309.

Wilson, A. C., R. Roelofs, M. Stern, N. Srebro, and B. Recht, 2017, The Marginal Value of Adaptive Gradient Methods in Machine Learning, in I. Guyon, U. V Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., Advances in Neural Information Processing Systems 30: Curran Associates, Inc., 4148–4158.

Xia, G.-S., J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, L. Zhang, and X. Lu, 2017, AID: A Benchmark Data Set for Performance Evaluation of Aerial Scene Classification: IEEE Transactions on Geoscience and Remote Sensing, 55, 3965–3981, doi:10.1109/TGRS.2017.2685945.

Xiao, Y., and Q. Zhan, 2009, A review of remote sensing applications in urban planning and management in China, in 2009 Joint Urban Remote Sensing Event: 1–5, doi:10.1109/URS.2009.5137653.

Yang, Y., and S. Newsam, 2010, Bag-of-visual-words and spatial extensions for land-use classification, in Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '10: ACM Press, 270, doi:10.1145/1869790.1869829.

Yang, X., Y. Ye, X. Li, R. Y. K. Lau, X. Zhang, and X. Huang, 2018, Hyperspectral Image Classification With Deep Learning Models: IEEE Transactions on Geoscience and Remote Sensing, 56, 5408–5423, doi:10.1109/TGRS.2018.2815613.

Yin, X., W. Chen, X. Wu, and H. Yue, 2017, Fine-tuning and visualization of convolutional neural networks, in 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA): IEEE, 1310–1315, doi:10.1109/ICIEA.2017.8283041.

Yosinski, J., J. Clune, Y. Bengio, and H. Lipson, 2014, How transferable are features in deep neural networks? Advances in Neural Information Processing Systems, 27, 3320–3328.

Zhou, W., S. Newsam, C. Li, and Z. Shao, 2018a, PatternNet: A benchmark dataset for performance evaluation of remote sensing image retrieval: ISPRS Journal of Photogrammetry and Remote Sensing, 145, 197–209, doi:10.1016/J.ISPRSJPRS.2018.01.004.

Zhou, W., S. Newsam, C. Li, and Z. Shao, 2018b, PatternNet: A benchmark dataset for performance evaluation of remote sensing image retrieval: ISPRS Journal of Photogrammetry and Remote Sensing, 145, 197–209, doi:10.1016/J.ISPRSJPRS.2018.01.004.

Chapter 6: Forecasting Megaelectron-Volt Electrons inside Earth's Outer Radiation Belt: PreMevE 2.0 Based on Supervised Machine Learning Algorithms

Rafael Pires de Lima[1,2], Yue Chen[1], Youzuo Lin[1],

[1]Los Alamos National Laboratory, Los Alamos, NM, USA,

[2]School of Geology and Geophysics, The University of Oklahoma, 100 East Boyd Street, RM 710, Norman, Oklahoma, 73019, USA

Preface.

In the last chapter of my dissertation I present some of the research I developed during my internship at the Los Alamos National Laboratory. The work in chapter 6 aims to forecast relativistic electrons inside Earth's outer radiation belt. Relativistic electrons are high energy and have the potential to destroy satellite electronics. I use different supervised learning models to forecast the behavior of 1.0 Megaelectron-volt (MeV) trapped inside Earth's outer Van Allen belt. Interestingly, the results show how linear models are well capable to model most of the 1.0 MeV fluctuation. The work presented in this chapter was submitted to a space weather journal and the preliminary results are published as abstract in Pires de Lima et al. (2019a). The pre-print version is already available (Pires de Lima et al., 2019b). Unrelated to space weather, the work I developed during my internship at Los Alamos National Laboratory using simulation data for CO2 capture studies was published as two expanded abstracts (Pires de Lima et al., 2019c; Pires de Lima and Lin, 2019).

References

Pires de Lima, R., Chen, Y., Lin, Y., 2019a. PreMevE 2.0: Neural Network Based Predictive Model for MeV Electrons in Earth's Outer Radiation Belt, in: AGU Fall Meeting 2019. American Geophysical Union (AGU).

Pires de Lima, R., Chen, Y., Lin, Y., 2019b. Forecasting Megaelectron-Volt Electrons inside Earth's Outer Radiation Belt: PreMevE 2.0 Based on Supervised Machine Learning Algorithms. https://doi.org/ArXiv: physics.space-ph/1911.01315

Pires de Lima, R., Lin, Y., 2019. Geophysical data integration and machine learning for multi-target leakage estimation in geologic carbon sequestration, in: SEG Technical Program Expanded Abstracts 2019. pp. 2333–2337. https://doi.org/10.1190/segam2019-3215405.1

Pires de Lima, R., Lin, Y., Marfurt, K.J., 2019c. Transforming seismic data into pseudo-RGB images to predict CO2 leakage using pre-learned convolutional neural networks weights, in: SEG Technical Program Expanded Abstracts 2019. Society of Exploration Geophysicists, pp. 2368–2372. https://doi.org/10.1190/segam2019-3215401.1

Abstract

Here we present the recent progress in upgrading a predictive model for Megaelectron-Volt (MeV) electrons inside the Earth's outer Van Allen belt. This updated model, called PreMevE 2.0, is demonstrated to make much improved forecasts, particularly at outer Lshells, by including upstream solar wind speeds to the model's input parameter list. Furthermore, based on several kinds of linear and artificial machine learning algorithms, a list of models was constructed, trained, validated and tested with 42-month MeV electron observations from Van Allen Probes. Out-of-sample test results from these models show that, with optimized model hyperparameters and input parameter combinations, the top performer from each category of models has the similar capability of making reliable 1-day (2-day) forecasts with Lshell-averaged performance efficiency values ~ 0.87 (~0.82). Interestingly, the linear regression model is often the most successful one when compared to other models, which indicates the relationship between 1 MeV electron dynamics and precipitating electrons is dominated by linear components. It is also shown that PreMevE 2.0 can reasonably predict the onsets of MeV electron events in 2-day forecasts. This improved PreMevE model is driven by observations from longstanding space infrastructure (a NOAA satellite in low-Earth-orbit, the solar wind monitor at the L1 point, and one LANL satellite in geosynchronous orbit) to make high-fidelity forecasts for MeV electrons, and thus can be an invaluable space weather forecasting tool for the future.

1. Introduction

Man-made satellites operating in medium- and high-altitude Earth orbits are continuously exposed to hazardous space radiation originated from different sources. Among them, one major contributor is the relativistic electron population—with energies comparable to and/or larger than their rest energy of 0.511 Megaelectron-volt (MeV)—trapped inside Earth's outer Van Allen belt. Owning to their high penetration capability, these MeV electrons are difficult to be fully stopped by normal shielding. Particularly, during MeV electron events when electron intensities across the outer belt are greatly enhanced to sustaining high levels, space-borne electronic systems with inadequate hardening are susceptible to deep-dielectric charging and discharging phenomenon caused by those electrons (Lai et al., 2018), and thus may suffer severe damage or even stop functioning. Therefore, protecting critical space infrastructures from harsh space weather conditions– including MeV electron events – has high priority for stakeholders such as the space industry, service providers and government agencies.

Similar to terrestrial weather services, real-time monitoring and model forecasting are the two principle ways of mitigating risks from outer-belt MeV electrons. Given the successful NASA Van Allen Probes mission, previously known as RBSP (Mauk et al., 2013), quickly approaches its end, the need of reliable forecasting models for MeV electrons becomes compelling once again due to the coming absence of in-situ measurements. Indeed, forecasting models have been developed including such as SPACECAST framework (Horne et al., 2013) for the whole outer radiation belt, and Relativistic Electron Forecast Model (based on Baker et al. (1990)) currently operated by NOAA specifically for electrons at geosynchronous (GEO) orbit. Recently, Chen et al. (2019) has developed and verified a new predictive MeV electron model called PreMevE to forecast MeV electron events throughout the whole outer radiation belt, using linear filters with inputs from low-Earth-orbit (LEO) observations. In this work, we further improve PreMevE model by applying and testing several supervised machine learning algorithms with optimized selection of input parameters.

Machine learning (ML) has been a topic in consideration for more than half a century (e.g., Minsky, 1961; Hartigan & Wong, 1979; Hopfield, 1982), and its popularity increased significantly in the last decade with numerous applications in various research fields. Examples of success include seismicity studies (e.g., Kortström et al., 2016; Perol et al., 2018; Sinha et al., 2018; Ren et al., 2019; Wang et al., 2019), geological mapping (e.g. Cracknell & Reading, 2014; Pires de Lima & Marfurt, 2018), optical/electrical geoscientific images classification (e.g. Duarte-Coronado et al., 2019; Pires de Lima et al., 2019a; Pires de Lima et al., 2019b; Valentín et al., 2019), medical image segmentation and classification (e.g., Ronneberger et al., 2015; Tajbakhsh et al., 2016; Qayyum et al., 2017), speech recognition (e.g., Graves & Schmidhuber, 2005; Graves et al., 2013), and etc. Among them, as observed by LeCun et al. (2015), the work of Krizhevsky et al. (2012) was the breakthrough responsive for the rapid adoption of deep learning by the computer vision and others communities.

Meanwhile, the application of ML also has gained momentum in the space weather community. An early use of artificial neural networks to predict the flux of energetic electrons at geosynchronous (GEO) orbit was presented by Stringer et al. (1996) in which GOES-7 data were used to make one-hour nowcasts of hourly-averaged fluxes of electrons at energies of 3-5 MeV. Later, Ukhorskiy et al. (2004) and Kitamura et al. (2011) used artificial neural networks to develop one-day forecasts of daily averaged electron fluxes at GEO. More recently, Shin et al. (2016) used a neural network scheme with solar wind inputs to predict GEO electrons over a wide energy range and different time resolutions. Wei et al. (2018) also successfully improved the one-day forecasts of >2 MeV electron fluxes at GEO by applying deep learning algorithms. For a review, Camporeale (2019) has summarized the recent progresses and opportunities of applying ML for space weather forecasting problems, including predicting geomagnetic indices, relativistic electrons, solar flares occurrence, coronal mass ejection propagation time, solar wind speed and etc.

The purpose of this work is to present how PreMevE has been upgraded with ML algorithms to make improved predictions of MeV electron flux distributions. With no requirement of in-situ MeV electron measurements except for at GEO, this unique model has shown its great potential of meeting the predictive requirements for outer-belt electrons during the post-RBSP era. Section 2 briefly describes the data and parameters used for this study, and the selected ML algorithms and their implementations are explained in Section 3. Section 4 compares and summarizes the prediction performance of different models, followed by detailed discussions in Section 5. This work is concluded by Section 6 with a summary of our findings and possible future directions.

2. Data and Input Parameters

Electron data used in this work include observations made by particle instruments aboard a RBSP spacecraft, one Los Alamos National Laboratory (LANL) GEO satellite, and one NOAA Polar Operational Environmental Satellite (POES) in a period ranging from February 2013 to August 2016, as shown in Figure 24. Electron data used here are the same as in Chen et al. (2019) in which detailed descriptions of the original data and their preparation can be found, and here is a brief recap. First, trapped 1 MeV electrons across a range of L-shells ($L \leq 6$) are in situ measured by the Magnetic Electron Ion Spectrometer (MagEIS) instrument (Blake et al., 2013) on board RBSP-a, and the spin-averaged fluxes are plotted in Panel A as a function of Lshell and time. Here we use McIlwain's L values (McIlwain, 1966) calculated from the quiet Olson and Pfitzer magnetic field model (Olson & Pfitzer, 1977) together with the International Geomagnetic Reference Field model. At GEO, we use observations from the Synchronous Orbit

Particle Analyzer (SOPA; Belian et al., 1992) instrument carried by the GEO satellite LANL-01A. For simplicity, all GEO fluxes are put on the fixed L = 6.6 and plotted in the top of Panel A. Then, precipitating electrons are monitored by the Space Environment Monitor 2 (SEM2) instruments on board NOAA POES satellites in low-Earth-orbits (LEOs, Evans et al., 2000), and the count rates from the 90° telescopes on NOAA-15 are presented for three energy channels as in Panels B, C and D. Here L values for NOAA-15 are calculated from the International Geomagnetic Reference Field model. Additionally, upstream solar wind (SW) speeds in Panel E are downloaded from CDAweb site and added to models' inputs. All RBSP-a, LANL-01A, and POES-15 electron fluxes as well as solar wind speeds in Figure 24 are binned by 5 hours to allow for RBSP's full coverage on the outer belt for each time bin. Lshell bin size for electrons is 0.1.

Throughout this work, we refer to POES electron fluxes at > 100 keV, > 300 keV, and > 1000 keV as E2, E3, and P6 respectively. Logarithmic values of E2, E3, and P6, along with standardized scaled values of SW speeds form the input data sets, or the predictors, being used to forecast the logarithm of 1 MeV trapped electron fluxes, sometimes also referred to as "target". The standardized of SW is done by subtracting the mean and dividing by the standard deviation (both the mean of 404.8 km/s and standard variation of 86.8 km/s are computed with the training set as defined in Section 4). Hereinafter, when we refer to 1 MeV target, E2, E3, and P6 fluxes, we are actually referring to their logarithmic values. Lshell coverage of this study is confined to 2.8 – 6 (the range of RBSP) and 6.6 (LANL GEO), while fluxes at other Lshells can be derived by interpolation or extrapolation (Chen et al., 2019).

3. Supervised Learning Algorithms

ML can be described as a collection of techniques in which systems improve their performance through automatic analysis of data. The power of ML models lies in their capacity to extract statistical information (patterns and features) from ample data with no requirement of hypothesis, in a sharp contrast to physics-based models in which researchers manually select parameters to be used as input for models with specific governing physics. ML models are capable of extracting signature and correspondence that might be overlooked by traditional methods, e.g., nonlinear relationship, and can be relatively easier to use with multiple input sources. Therefore, under certain circumstances, ML models can outperform traditional ones. For example, Tajbakhsh et al. (2016) found that deep neural network models outperformed handcrafted solutions in medical image analysis tasks. One of major drawbacks of ML models, particularly deep neural networks, is its incomplete capability in interpretability ("how") and explainability ("why") (Murdoch et al., 2019). Thus, sometimes ML models can be complicated to explain, hindering our ability to propose new theories based on ML results.

Common ML algorithm types include supervised, unsupervised, semi-supervised, and reinforcement learning (Ayodele, 2010). The algorithms used here fall under the category of supervised learning as they make use of input sample data paired with an appropriate label. The label here refers to 1 MeV electron flux at different Lshells, the target value to be forecasted. Moreover, the models implemented here can be classified as regressions, as the labels are specified scalar values.

As explained by Camporeale (2019), supervised regressors try to find the mapping relationship between a set of multidimensional inputs $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ and its corresponding scalar output label y, under the general form

$$y = f(\boldsymbol{x}) + \epsilon, \quad (1)$$

where $f: \mathbb{R}^N \rightarrow \mathbb{R}$ is a linear or nonlinear function and ϵ represents additive noise. All methods used to find the unknown function f can be seen as an optimization problem, where the objective is to minimize a given loss function. The loss function is a function that maps the distance between all the predicted and target values into a real number, therefore providing some "cost" associated with the prediction. The following four subsections provide more details in each one of the supervised regressor models used in this study. A comprehensive discussion on artificial neural networks and deep learning models can be found in LeCun et al. (2015), with information about techniques common to several artificial intelligence applications. To exemplify the supervised learning problem as a flux forecasting task, consider predicting the 1 MeV electron fluxes at time t at GEO shell using the past values of 1 MeV electrons at GEO. Suppose we use M training samples to perform the analysis, and the number of past values we wish to use for each time step is four ($N = 4$). That is, we have M pairs of $(\boldsymbol{x}_t, y_t)$ training samples, or $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \ldots, (\boldsymbol{x}_M, y_M)\}$ where $\boldsymbol{x_t} = (x_{t-1}, x_{t-2}, x_{t-3}, x_{t-4})^T \in \mathbb{R}^{N=4}$ and $y_t \in \mathbb{R}$. We can rewrite the predictors $\boldsymbol{x_t}$ as a matrix $X \in \mathbb{R}^{NxM}$, where each column of the matrix X represents one $\boldsymbol{x_t}$ training sample vector**.** The y_t samples can also be defined as a single row matrix $Y \in \mathbb{R}^{1xM}$. The goal of ML training is to optimize the internal parameter values of the given mapping function *f*—a specified ML algorithm—by minimizing the loss function associated with the noise matrix ϵ after inserting *X* and *Y* back into Eq. (1). Here we use the past values of multiple input data, including E2, E3, P6, and solar wind speed, to forecast 1 MeV electron fluxes at each individual L-shell. Next, we describe the four selected algorithms including linear regression, multilayer perceptron, convolutional neural networks, and long short-term memory methods.

3.1 Linear Regression

Linear regression is the simplest supervised learning method, while sometimes it is also interpreted as the simplest ML algorithm. This algorithm has a vast range of applications and constitutes a basic building block for more complex algorithms. The linear regression equation is given by

$$f(\boldsymbol{x_i}) = \boldsymbol{w^T x_i} + \boldsymbol{b}, \tag{2}$$

where $\boldsymbol{w}$ is a vector containing weights and $\boldsymbol{b}$ is the bias term. In a predictive problem, y as in Eq. (1) represents the label, or target, to be predicted (the 1 MeV electron flux), $\boldsymbol{x}$ represents the input data (e.g. past values of precipitating electron fluxes) and $\boldsymbol{w}$ represents the set of linear coefficients that minimize the loss, or the sum of the errors of all true values of y and the predicted $f(\boldsymbol{x_i})$. From the optimization perspective, the weights $\boldsymbol{w}$ can be obtained using a simple ordinary least squares method. Linear models are simple models generally very useful as baselines, and their selection for this work is also due to the success of previous work by Chen et al. (2019).

3.2 Multilayer Perceptron

Starting from the linear model, a single neuron can be defined as

$$f = a(\boldsymbol{w^T x_i} + \boldsymbol{b}), \tag{3}$$

where $a(.)$ is an element-wise activation function. The activation function is responsible to introduce non-linearity to the model. Some of the most common activation functions are the Rectified Linear Unit (ReLU, Hahnloser et al., 2000; Nair & Hinton, 2010) and the Exponential

Linear Unit (ELU, Clevert et al., 2015). ReLU is a piecewise linear function that outputs the input for positive values, zero otherwise. ELU outputs the identity for positive values as well, however ELU uses a logarithm curve for negative values ($constant(exp(input) - \mathbf{1})$). A hidden layer is a set of neurons, or units, that take in a set of inputs ($\boldsymbol{x}$) and produce an output f. If we use the $f_i^{[l]}$ notation to represent the output of the neuron i at layer l, we can write Eq. (3) for the following layer as $f^{l+1} = a\left(\boldsymbol{w^T f^{[l]}} + \boldsymbol{b}\right)$ to represent the inputs for layer l +1 depend on the output of layer l. Figure 25 illustrates a single neuron in the left and how sets of neurons can be combined to form layers and neural networks in the right. Here the information flows from left (the input) to right (the output). This structure is a class of Feedforward Networks, sometimes named multilayer perceptron (MLP). Loosely defined, an artificial neural network (NN) is a model consisting of connected neurons. The term deep model or deep learning is generally used for neural networks containing more than one hidden layer. When all neurons in a layer receive input from all elements in the previous layer (e.g. the hidden layers in Figure 25b), they are also called fully or densely connected layers.

3.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are powerful and influential deep learning model architectures. The computer vision field strongly adopted CNNs as their workforce after the CNN described in Krizhevsky et al. (2012) and achieved new levels of accuracy in the popular ImageNet Large Scale Visual Recognition Competition (Russakovsky et al., 2015). All CNNs make use of the fundamental convolutional kernel. Convolution operates on two functions, one generally interpreted as the “input”, and the other as a “filter”. The filter is commonly referred to as “kernel”. The kernel is applied on the input, producing an output image or signal. During the training stage, the values of kernels are updated in such a way that the output generated by the CNN is more similar to the desired label, i.e., minimizes the cost. Just like the neurons described in subsection 3.2, a set of convolutional kernels can be combined into layers. Dumoulin & Visin (2016) showed details on the arithmetic of convolutions for deep learning. Here, we provide only the essential equation for 1D convolution. An 1D convolution of the input vector $\boldsymbol{x}$ and the kernel $\boldsymbol{g}$ of length m is given by

$$(x * g)(i) = \sum_{j=1}^{m} g(j)x\left(i - j + \frac{m}{2}\right). \tag{4}$$

A CNN unit in deep learning models is a composite of activation function and the convolution term in Eq. (4), i.e., $f(i) = a((x * g)(i))$.

Springenberg et al. (2014) observed that CNNs commonly use alternating convolution and max-pooling layers followed by a small number of fully connected layers. The models are typically regularized during training using dropout. Max-pooling are simple down-sampling steps in which the maximum value for each patch (containing multiple values) of a feature is used to represent the entire patch, effectively reducing the feature size. Dropouts layers randomly select a percentage of their inputs to be ignored during the training phase. Dropouts are useful to avoid overfitting. Dropout is a general approach and not specific for CNN models. Srivastava et al. (2014) showed that dropout improves the performance of neural networks on many supervised learning tasks such as speech recognition, document classification, vision and computational biology.

3.4 Long Short-term Memory

Long short-term memory (LSTM) networks are a popular recurrent neural network (RNN) structure introduced by Hochreiter & Schmidhuber (1997). RNN is a class of artificial neural networks in which neurons can be connected to form a directed graph along a temporal sequence (Figure 26). Different from traditional feedforward NNs, LSTM has internal loops to allow to retain information from previous times and decide its usage for predictions. Indeed, the LSTM basic unit is called memory cell inside which internal components can decide when to keep or override information in the memory cell, when to access the information in memory cell, and when to prevent other units from being perturbated (Hochreiter & Schmidhuber, 1997). Olah (2015) provides a detailed walkthrough of the LSTM components. LSTMs are constantly used in speech recognition problems (e.g. Graves et al., 2013; Graves & Schmidhuber, 2005) as well as forecasting (e.g. Kong et al., 2019). Here LSTM was selected for testing as a representative of RNNs.

4. Testing Algorithms and Model Performance

Following ML best practices, we split the data into training, validation, and test sets. The training set is the data effectively used for model optimization. The validation set is used to tune model hyperparameters, such as the number of neurons/layers or optimization options. Finally, the test set is reserved for model performance evaluation on the final stage. Here, the training data set consists of observations in the first 4,008 time bins (roughly 835 days, or 27.4 months, 65% of the whole data set), the validation set has observations for the next 841 time bins (roughly 175 days, or 5.8 months, 14% of the data), and the test set is for the final 1,280 time bins (roughly 267 days, or 8.8 months, 21% of the data). Observational data are split in such a manner so that the major observational gap over days 840 – 850 is in between the sets, thus the models are always trained, validated, and tested in segments containing real continuous observations.

The optimization goal for all the models is to reduce the root-mean-square error (RMSE) between the real value $\boldsymbol{y}$ and the predicted value $\boldsymbol{f}$, both with the size M. RMSE is defined as $\sqrt{\frac{\sum_{j=1}^{M}(f_j - y_j)^2}{M}}$. In this study, linear models minimize the error using the ordinary least squares, while artificial neural network models use Adam optimization as defined by Kingma & Ba (2014).

Chen et al. (2019) has demonstrated that E2 fluxes can be used for predicting the onset timing of MeV electron events, and here we also computed the normalized temporal derivatives of E2 fluxes, naming it dE2, and tested by adding it to the input data sets. The dE2 at time bin *t* for E2 is defined as $dE2_t = \frac{E2_t - E2_{t-1}}{E2_{t-1}}$. The temporal correlation between E2, dE2, and trapped MeV electron fluxes can be recognized from Figure 27.

4.1 Test Input Parameter Combinations

Our first experiment tests different combinations of input parameters with the objective to find the set of input data that can best predict 1 MeV electrons. Specifically, we use Linear and LSTM models to evaluate what combination of input parameters yields the highest Performance Efficiency (PE). PE provides a measure of quantifying the accuracy of predictions by comparing to variance of the target. Naming y as the true value (the logarithm of the target 1 MeV electron flux) and f as the predicted value, both with size M, PE is defined as

$$PE = 1 - \frac{\sum_{\boldsymbol{j=1}}^{\boldsymbol{M}} \left(y_j - f_j\right)^{\boldsymbol{2}}}{\sum_{\boldsymbol{j=1}}^{\boldsymbol{M}} \left(y_j - \bar{y}\right)^{\boldsymbol{2}}}, \tag{5}$$

where $\bar{y}$ is the mean of $\boldsymbol{y}$. PE does not have a lower bound, and the perfect score is 1.0, meaning all predicted value perfectly match observed data, or that $\boldsymbol{f} = \boldsymbol{y}$.

To make 1-day (25 hours) forecasts of MeV electrons for a single Lshell, our models ingest the past values of the input data at the same L-shell. The only exception is at GEO, where the model inputs include the MeV past of GEO from measurements. Additionally, as Chen et al. (2019) found E2, E3, and P6 values at GEO have relatively weak correlations with 1 MeV electrons, E2, E3, and P6 channels at L-shell of 4.6 are used instead for model inputs. The term “window size” refers to how many five-hour time bins of input data are needed by the models. Chen et al. (2019) found a window size of 15 time bins (equivalent to 75 hours) to be effective for the forecast of MeV electrons. Adhering to the “power of two” ML convention, we used a window size of 16. The “power of two” rule is based on the fact that CPUs and GPUs memory architecture are usually organized in powers of two, thus using power of two data organization can be beneficial for computation efficiency. For naming convention, when a LSTM model has one layer with 128 memory cells, we use LSTM-128 as the name for this model; the linear models are referred to as LinearReg throughout the manuscript. Results from the submodel 1 and 2 of previous PreMevE in Chen et al. (2019) are cited as linear1 and linear2 for a baseline comparison through the manuscript. Note in this work all PE values and fluxes from linear1 and linear2 are for 1-day forecasts only.

Table 17 summarizes the overall PE values (averaged over all Lshells) for twenty tests performed for 1-day predictions. For each of the two categories of models, ten input parameter sets are tested, starting from each single parameter to various combinations. Here we focus on the out-of-sample PE values, i.e., those in the column of PE val+test, to judge model performance. The general trend is that more parameters lead to better performance. For example, the last LinearReg model with all parameters as input (the 10^{th} model) has not only the overall highest PE value of 0.861 but also at GEO (0.587). These two values are higher than those for linear2 (0.797 and 0.352), which indicates significant improvements. (The linear1 was designed for capturing onset timings of MeV electron events and thus its PE is always lower than that of linear2 (Chen et al., 2019).) Interestingly, the last two LSTM models (19^{th} and 20^{th}) have the highest overall PE values for this category, but still slightly lower than those of the 10^{th} LinearReg model.

In this step, we also confirmed that adding SW speeds to the input list improves model performance, which was not tested previously in Chen et al. (2019). In Table 17, the overall PE for the 1^{st} model by using SW speed as the sole input parameter is 0.518, which indicates this simple model can predict MeV electrons over the whole outer belt to some degree but not as well as the linear2, although the PE of 0.557 at GEO is much higher than that of linear2 (0.352). In comparison, PE values from the 11^{th} model show that using SW speed as the sole parameter for LSTM model is not as good as for LinearReg particularly for GEO. When comparing models with and without SW speeds, e.g., the 2^{nd} vs 7^{th} (12^{th} vs 17^{th}) and 8^{th} vs 9^{th} (18^{th} vs 19^{th}), the improvements in overall PE are 0.009 (0.017) and 0.005 (0.013), while the improvements in PE at GEO are more significant up to 0.11. We also tested the dE2 and its addition to the input has not effects as significant as SW speeds when comparing the PE values of the 10^{th} (20^{th}) model to those of the 9^{th} (19^{th}).

Details of how model PEs improve as a function of Lshell are presented in Figure 28. For models in both categories, the top performer has much higher PE values than those of linear2 across the whole belt, with the most significant improvements at Lshells > ~ 4.5 and the maximum difference > 0.4 with L ~ 5.5. It can be clearly seen from the green curve in Panel A that SW speeds are a very helpful predictor for outer L-shells especially for LinearReg models, but inefficient for inner Lshells. This can be explained by the fact that in the large-Lshell region particle dynamics are more controlled by adiabatic effects, and is also consistent with the experience from predictive models for electrons at GEO (e.g., Baker et al., 1990). However, as in Panel B, the LSTM model using SW speeds as the sole predictor has only a few L-shells with PE values greater than zero. In summary, results in both Table 17 and Figure 28 suggest that the model PE values are higher with the use of more input data from multiple precipitating electron channels as well as the SW speeds. Therefore, tests in the rest of this study use the parameter combination including all inputs.

4.2 Model Selection and Metrics Evaluation

We then advanced to test a list of models built upon different algorithms with varying model hyperparameters (e.g., the window size and number of neurons). There are four different categories of models—Linear, MLP, LSTM, and CNNs—as described in Section 3, and here are how these models and test runs were set up. First, to account for cross-shell information as in Chen et al. 2019, some tests include E2 data at the L-shell of 4.6 as input for all other L-shells. Second, the MLP models presented here are composed of two hidden layers, the first one has 64 neurons, the second has 32 neurons, and the neurons use ELU as the activation function. In our early testing, we discovered ELU achieving marginally better performance than the most adopted ReLU activation function. A dropout layer that randomly selects 50% of the input to be inactivated after each one of the activation functions is included to help prevent overfitting. The output layer consists of a single neuron without an activation function. The dropout layer, used during training and deactivated during prediction, and the output layer are not accounted as hidden layers, but are also part of the model. We name such models MLP-64-32-elu. Then, CNN models with a window size 16 are composed of two convolutional layers, the first convolutional layer contains 64 kernels followed by a Max-pooling layer with size and stride equal two, and the second convolutional layer contains 32 kernels followed by a Max-pooling layer with the same size and stride. The CNN models with a window size 4 are composed of a single convolutional layer with 64 kernels followed by a Max-pooling layer. The kernels are one-dimensional with a size of three and are use ReLU as activation function. The convolutional layers are finalized with 50% dropout. The output layer consists of a single neuron without an activation function. Those CNN models are named Conv-64-32 and Conv-64, respectively. Finally, LSTM models follow the same structure as the ones described in Section 4.1.

Model performance is again evaluated by PE values by comparing forecasts to the target data. Table 18 presents the overall PE values for 24 test runs performed for 1-day predictions, and Table 19 presents PE values for the same test runs for 2-days predictions. Inside each category, the effects of window size, neuron/layer numbers and input parameters are tested and compared, and Table 17 and 18 only show results of models with good performance. For 1-day forecasts as in Table 18, the 6th LinearReg model has the high overall PE of 0.872 for out-of-sample test and 0.587 at GEO. Top models in the other three categories have similar overall and GEO PE values. All those values are higher than the overall PE of 0.797 and GEO PE of 0.352 from linear2 for 1-day forecasts. For 2-day forecasts in Table 19, top performers are the same as for 1-day predictions except for the MLP category. Here, the 6th LinearReg model has the highest overall PE of 0.827 for out-of-sample test and 0.333 at GEO. Again, top performers have overall PEs

~0.82 for 2-day predictions, which is lower than the ~0.87 for 1-day predictions but higher than the ~0.80 of linear2 for 1-day forecasts. (Chen et al. (2019) has shown that linear1 and linear2 have lower PE for 2-day forecasts than 1-day.) Their PEs at GEO are mostly above 0.33, comparable to linear2. Note for the MLP category, the 9th model is the top performer in Table 19, with no E2 at L=4.6 for input—instead of the 11th in Table 17. All CNN models in Table 19 cannot make 2-day forecasts at GEO very well.

Figure 29 plots PE curves for both 1- and 2-day forecasts as a function of Lshell, which further confirm our models' performance are more robust than previous results published in Chen et al. (2019). First, the PE curves for all four top models cluster together, with values at outer L-shells (minimum > ~0.3) lower than those at inner L-shells (maximum > 0.8 in left panel and >0.7 for right). All PE curves for both 1-day and 2-day are well above that from linear2 (1-day) expect at low Lshells for 2-day forecasts. The most significant improvements in PE are at Lshells > 4.5. For 1-day forecasts, the 6th model of LinearReg in Table 18 (green thick line in Figure 29A) can be seen to outperform with higher overall PE than the 10th model in Table 17 with the addition of E2 at L=4.6 to the parameter list. In addition, the performance of LinearReg models is persistently good for both 1- and 2-day forecasts, particularly at GEO where other top models degrade quickly.

It is striking how the models (LinearReg, MLP, LSTM, CNN) show very similar forecasting ability when using similar input data. Plus, the linear models seem to have leading performance for the forecasting in many scenarios, particularly for 2-day predictions. Two main observations should be taken for such behaviors. The first one is that a great part of the interplay between trapped 1 MeV electrons and input parameters (precipitating electrons and SW speeds) appear to be mainly linear. Previous PreMevE in Chen et al. (2019) has shown high PE using linear filters to forecast MeV electrons, and our findings corroborate previous results. The second observation is that artificial NNs, as depicted in Section 3, have their linear component. As a linear model achieves good results, artificial neural networks are expected to do at least the same. Thus, the dominance of linear components explains why the top models from all four categories of algorithms have very similar predictive performance. In addition, the secondary role from non-linear components make CNN models having the best overall PE for validation and test set combined in Table 18, as well as MLP and LSTM models having the best PE at GEO (Tables 18 and 19). Therefore, this new PreMevE 2.0 model will indeed include all four algorithms, which form an ensemble of predictive models whose relative weights are left to future work. In next, we take a closer look at predicted results from all four algorithms.

5. Detailed Predictions and Discussions

An overview of the 1-day forecasted flux distributions is exhibited in Figure 30 compared to the 1 MeV flux target. Visually, forecasted distributions from the four top performers as in Table 18 (Panels B-D) resemble the observations (Panel A) very closely. Portions of data used for training, validation, and test are marked out by color bars in the bottom of the figure. It can be seen that the flux enhancement, elevated flux levels (red regions), and decay afterwards during each individual MeV electron event are reproduced very well, although the dropouts of MeV electrons (blue strips) at large Lshells are often not well captured (e.g., the one on ~ day 1080) or even totally missed (e.g., the one on ~ day 870). It is deemed acceptable at this stage since PreMevE model mainly aims to forecast high flux levels of MeV electrons. Similarly, Figure 31 compares 2-day forecasted results to target data and shows an akin resemblance, confirming the stable predictive performance of PreMevE 2.0 with a longer lead time.

Furthermore, Figure 32 shows more details how closely the 1-day forecasted fluxes are compared to the targets over combined validation and test period for selected L-shells. Here flux curves from the same models as in Figure 30 as well as linear2 model are plotted. The four PreMevE 2.0 model curves pack together tightly and follow the target curve closely, particularly during decays of high intensity events. The closeness between the target (black) and each forecasted curve depicts the performance of each model. A close inspection reveals that the linear2 curve (yellow) is often the one farthest away from the target, showing as almost the envelop line of the predictions, while the LinearReg curve (green) appears the closest tracer of target at L=4.5 and the MLP curve (red) is the winner for other two Lshells. Nevertheless, it can be seen that the forecasted values often lag behind the target at onsets of MeV electron events, e.g., the ones on ~ day 988 and 1093 at L = 4.5.

Figure 33 illustrates how well the onsets of MeV electron events at L=4.5 are captured by models. Here forecasts from linear1 is also plotted in blue for comparison. (Linear1, or the submodel 1, in Chen et al. (2019) is specifically designed to predict the onsets.) We selected 16 major events in which MeV electron fluxes increase by > ~10 times, marked out by the vertical gray boxes in Figure 33. Linear1 (the blue curve) successfully predicts the onsets of all major MeV electron events at this Lshell, indicated by the leading edges of significant sudden increments in fluxes fallen within the boxes with a width of 25 hours (also called prediction windows). In comparison, although the four models (particularly the LinearReg model in green) often predict onsets earlier than linear2, they only successfully predict eight of them (those marked with green letter Y), failed seven, and another barely making inside the prediction widow. In other words, 1-day forecasts from PreMevE 2.0 predict the onsets at L =4.5 with a success rate below 50%, which is better than linear2 but far behind linear1.

Two-day forecasts are also presented as in Figure 34 and 35. Again, forecasted results at three Lshells in Figure 34 closely trace the target, similar to Figure 32. Interestingly, for all 16 selected major MeV electron events in Figure 35, the onsets of 11 events are successfully predicted by the four models at L = 4.5, while the failed events decrease to 4. This increases the success rate of onset prediction to ~70%. From this sense, this new PreMevE 2.0 model is able to combine the advantages of both linear1 and linear2 by not only predicting the arrivals of new MeV electrons but also specifying evolving flux levels closely, which is a very encouraging progress.

Results from LinearReg and LSTM models at GEO are specifically presented in Figure 36 for both 1- and 2-day predictions. For 1-day forecasts in the top three panels, it can be seen that fluxes from LinearReg (green) and LSTM (purple) trace observations (black) more closely than linear2 (yellow), consistent with their higher PE values as shown in Table 18. Also, forecasts from LinearReg and LSTM appear to predict the onsets of MeV electron events about the same time as linear1, by comparing the leading edges of spikes of those flux curves. For 2-day forecasts, LinearReg PE value in Table 19 suggests that 2-day forecasts from LinearReg model are close to 1-day forecasts from linear2, which can be seen from entangled LinearReg and linear2 curves as in Panels D-F. Forecasts from the LSTM model is not as good, although they still capture the general trend of 1 MeV electrons at GEO.

Despite Chen et al.'s (2019) and our time window selection, the question of for how long history of each particle population affects each other remains open. Figure 37 shows the Spearman correlation of the input data (E2, E3, P6, SW speed) and the target (1 MeV electrons) for three selected L-shells using different time lags. Spearman correlation does not assume that the data follows a particular distribution, so it is a non-parametric measure of monotonic relationship.

The results in Figure 37 show that the Spearman correlation between input and target decays with longer time lags. The correlation remains stronger for longer periods in inner L-shells (i.e., longer memory) and decays faster for outer L-shells (shorter memory). We also note the correlation between SW and target gets more significant when moving to outer L-shells, which is consistent from our discussions in Section 4.1. Curiously, the shape the correlation curve of E3 is similar to the shape of P6, whereas the shape of E2 is similar to the shape of SW. All these suggest more robust models can be elaborated with a variation of inputs (window sizes and parameter combinations) for different L-shells. In fact, Figure 28 shows that E2+SW are apparently the best combination of input for outer shell prediction, whereas other combination of input presents stronger values of PE for inner shells. Given a threshold value of ~0.4 for significant correlation, it is seen from Figure 37 that a fixed window size for all Lshells may range from ~14 (to include the maximum correlation values) up to ~20 (to avoid too long history).

Previously Chen et al. (2019) used 300 time bins to train linear1 and linear2 models to forecast MeV electrons. Table 17 shows that linear1 and linear2 models have a weaker forecasting performance than the linear models trained with the similar inputs. The difference in performance can be explained by the fact that a much larger training set incorporate wider flux variations that can be helpful to train the models. Besides, the addition of SW speeds definitely helps improve the performance of linear models at large Lshells.

In this work, we have performed tests on number of units, types of activation functions, and number of layers, though it is still possible that a more intensive artificial NN architecture testing will find a more appropriate model for the MeV forecasting. Moreover, we hypothesize that more data can be useful to improve models' performance. We plan to test with observations over longer period as well as for higher energy electrons in the next step.

6. Summary and Conclusions

This new PreMevE 2.0 model aims to forecast MeV electron distributions even with no in-situ measurements available, e.g., during the post-RBSP era, and it is designed to be driven by easily accessible inputs from long-standing satellite constellations in LEO and GEO as well as at the Lagrangian 1 point of Sun-Earth system. Meanwhile, deep learning algorithms have recently achieved new state-of-the-art accuracy in many problems partially due to the increase in available data. Therefore, it is reasonable for us to foresee an increase in both performance and use of deep learning model for MeV electron forecasting as more space weather data has been accumulated and made available.

In this work, we have tested (1) different model input parameter combinations and (2) four categories of supervised machine learning algorithms, aiming to upgrade our predictive model for MeV electrons inside the Earth's outer radiation belt. This new PreMevE 2.0 model has been demonstrated to make much improved forecasts, particularly at large Lshells, by including upstream solar wind speeds to the model's inputs. Additionally, based on four categories of linear and artificial machine learning algorithms, a list of models was constructed, trained, validated and tested with 42-month MeV electron observations from NASA Van Allen Probes mission. Model predictions over the 14-month long out-of-sample test show that, with optimized model hyperparameters and input parameter combinations, the top performer from each category of models has the similar capability of making reliable 1- and 2-day forecasts with

Lshell-averaged performance efficiency values of ~ 0.87 and ~0.82, respectively. Interestingly, the linear regression model is often the most successful one when compared to other models, which indicates the relationship between 1 MeV electron dynamics and precipitating electrons is dominated by linear components. It is also shown that PreMevE 2.0 can predict the onsets of MeV electron events in 2-day forecasts with a reasonable success rate of ~70%. This improved PreMevE model is driven by observations from longstanding space infrastructure (a NOAA LEO satellite, the solar wind monitor at L1 point, and one LANL GEO satellite) to make high-fidelity forecasts for MeV electrons, and thus can be an invaluable space weather forecasting tool for the community.

Figures

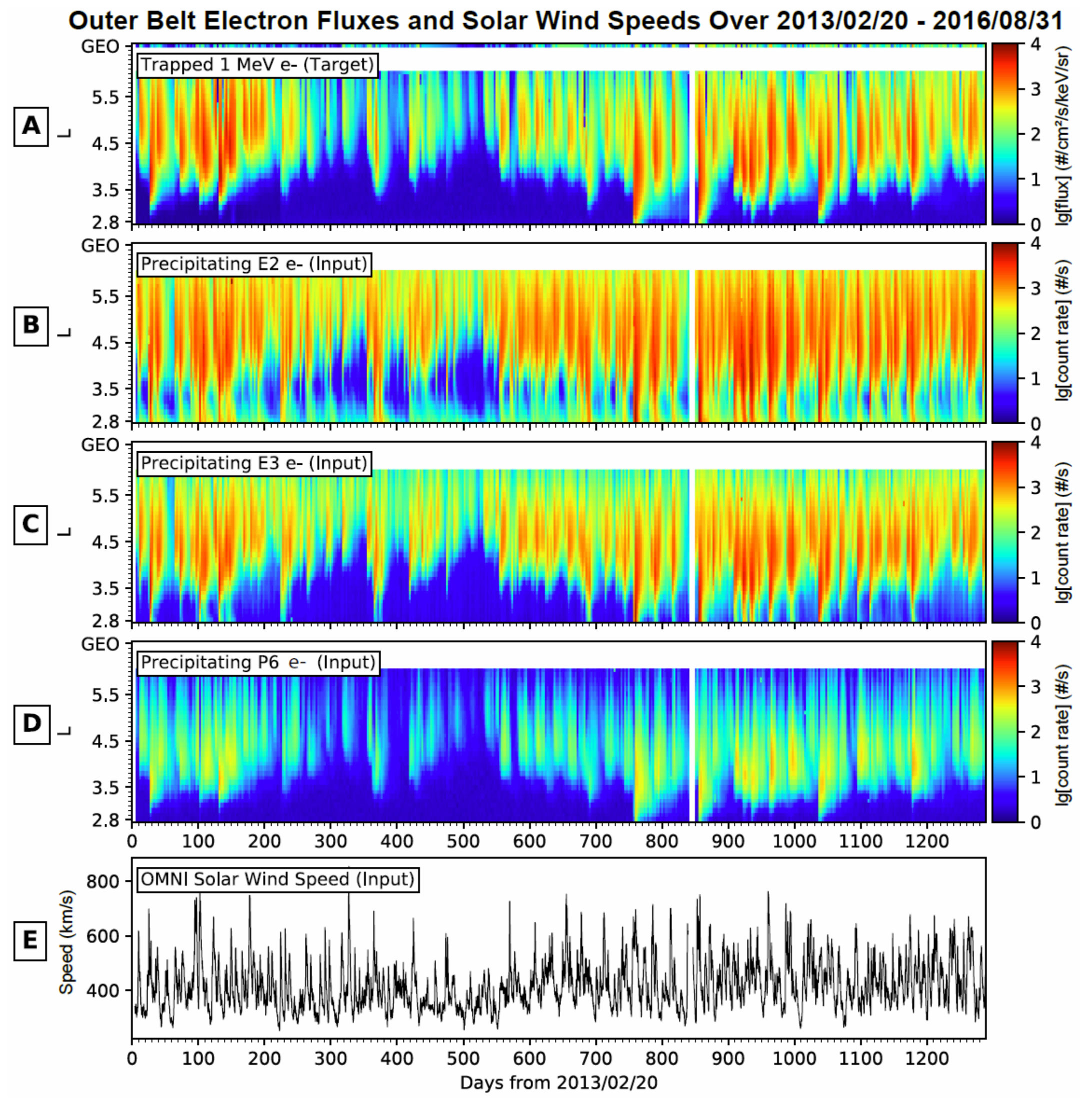


Figure 38: Overview of electron observations and solar wind speeds used in this study. All panels present for the same 1289-day interval starting from 2013/02/20. Panel A shows the flux distributions of 1 MeV electrons, the variable to be forecasted (i.e., targets). Similarly, **B**, **C**, and **D** show count rates of precipitating electrons measured by NOAA-15 in a low-Earth-orbit, for E2, E3, and P6 channels respectively. **E** plots the solar wind speeds upstream of the magnetosphere as in the OMNI data set for the period. Data in Panels B-E are model inputs (i.e., predictors).

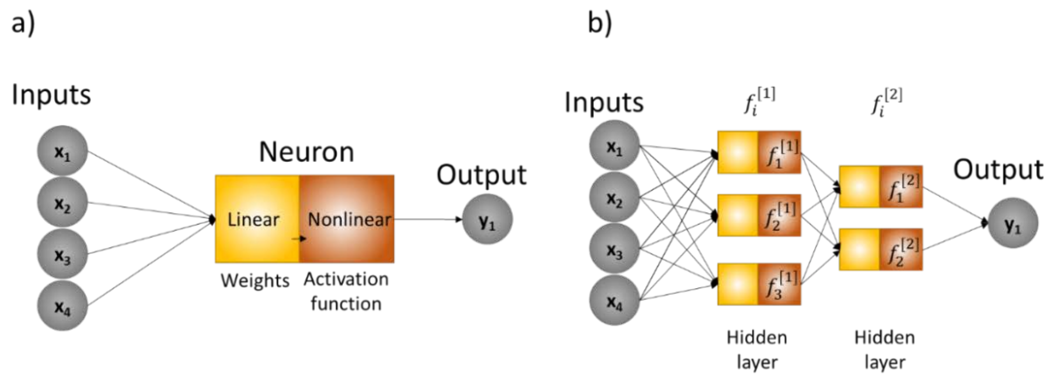


Figure 39: Visual generic representation of a single neuron and an artificial neural network. a) shows a single neuron that can be split into linear and nonlinear components, as well as the input and output data. In the case of a forecasting problem, the inputs can be data representing past times $t_{-1}, t_{-2}, t_{-3}, t_{-4}$, and the output is prediction at current time t_0 or even some future time. **b** shows how a set of neurons constitutes a layer and how the output of a layer can be used as input for the next layer.

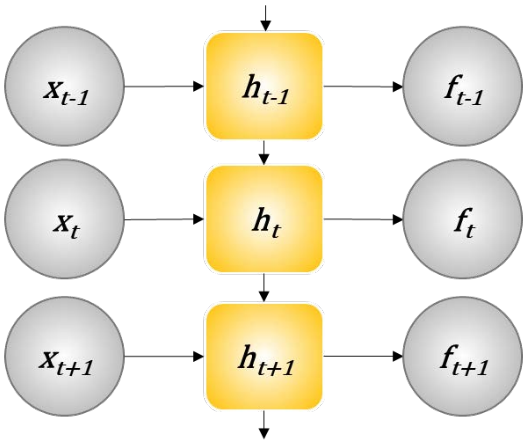


Figure 40: Representation of a recurrent neural network. In LSTM models, the basic unit $\boldsymbol{h}$ is also called a memory cell. The input vector $\boldsymbol{x}$ at an arbitrary time $\boldsymbol{t}$ is processed by a memory cell $\boldsymbol{h}$ and produces an output $\boldsymbol{f}(\boldsymbol{x})$. The output produced by $\boldsymbol{h_{t-1}}$ is also part of the input for $\boldsymbol{h_t}$. Thus, events at time $\boldsymbol{t}$ are processed with information from the previous steps. The output produced by $\boldsymbol{h}$ can be used as input to the next layer just like the described for the previous models.

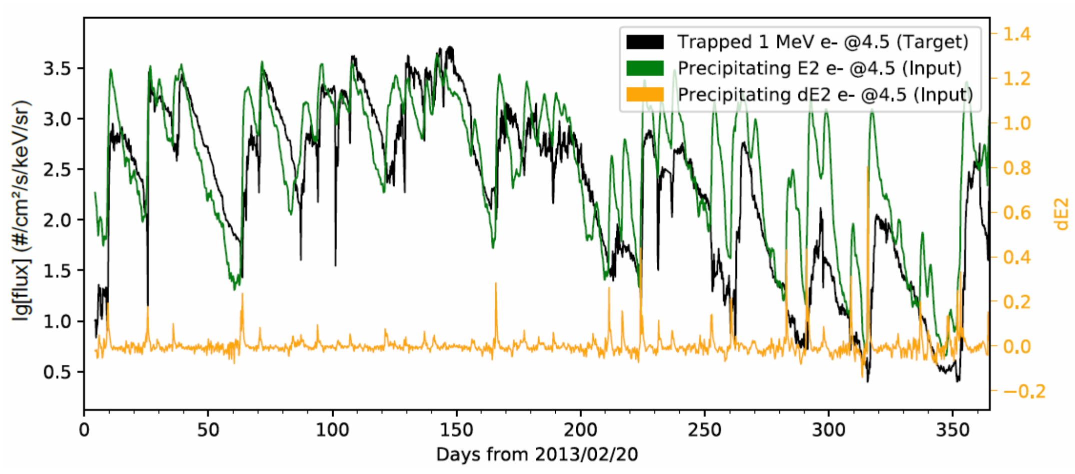


Figure 41. Temporal correlation between E2, dE2, and 1 MeV electrons fluxes in the first year of the interval. Note the leading edges of E2 increments (green) and the spikes in dE2 (yellow) generally precede the onsets of MeV electron events with a significant one-to-one temporal relationship.

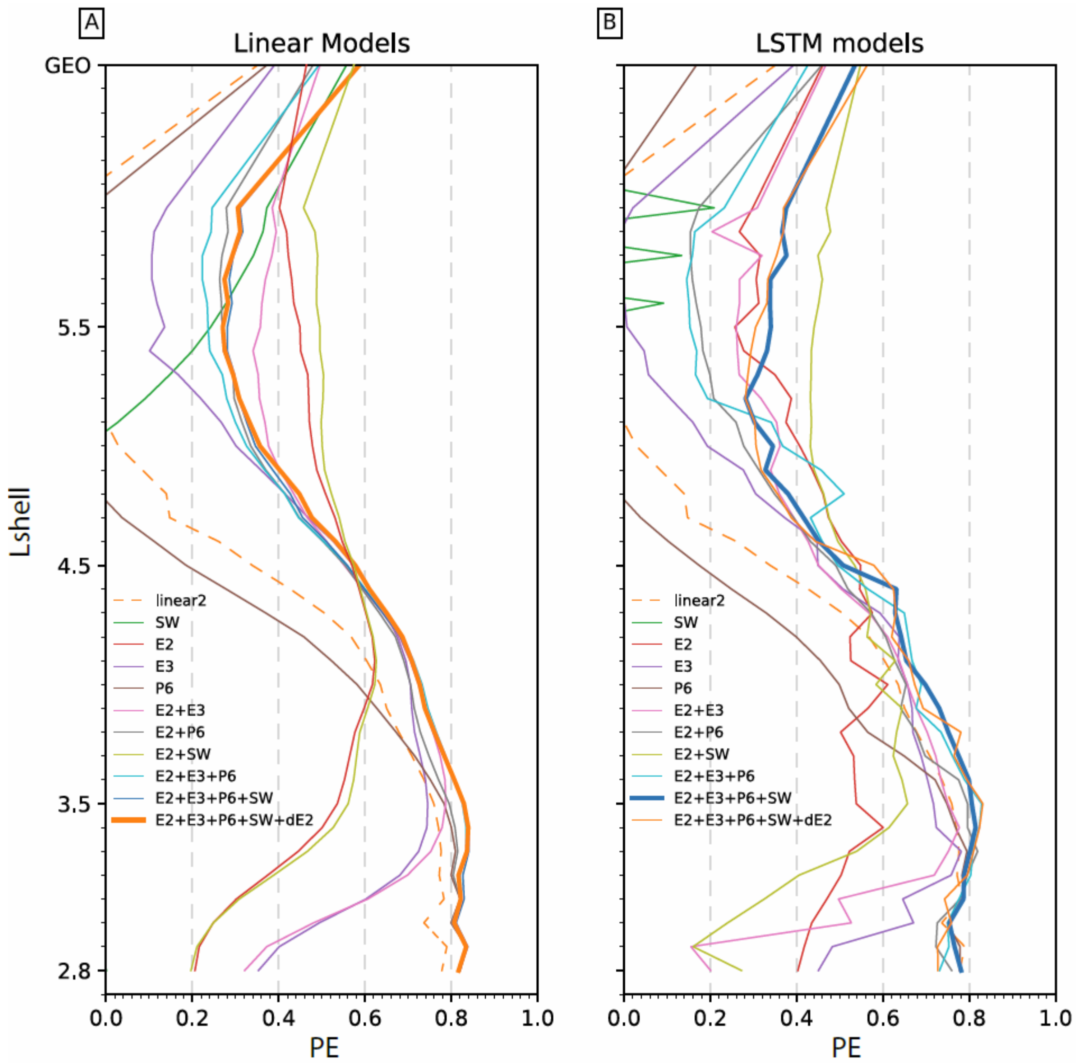


Figure 42. PE values for the combined validation and test sets are presented as a function of Lshell for different models and input parameters as in Table 17. **A.** LinearReg models and **B.** LSTM models. PE curve for linear2 model (dashed) is plotted for comparison. The model with the best performance—highest overall PE—for each category is highlighted with thick line.

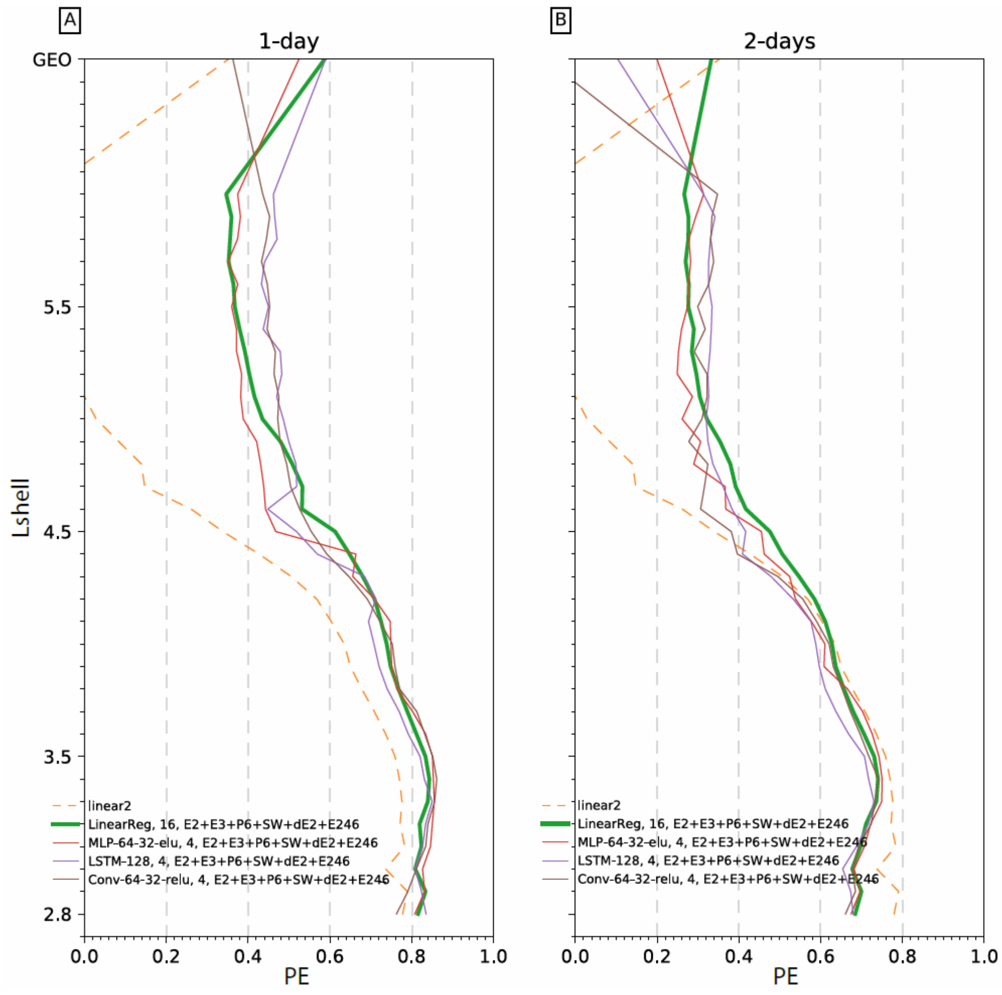


Figure 43. Model PE values over the combined validation and test data sets are presented as a function of Lshell for the top performers in Table 18 and 17. A. Top performer of each category for 1-day forecasts. PE curve of linear 2 for 1-day forecasts is plotted in dashed line for comparison. B. Top performer of each category for 2-day forecasts. Note the dashed line is still linear2 for 1-day forecasts. LinearReg models are highlighted in thick lines in both panels.

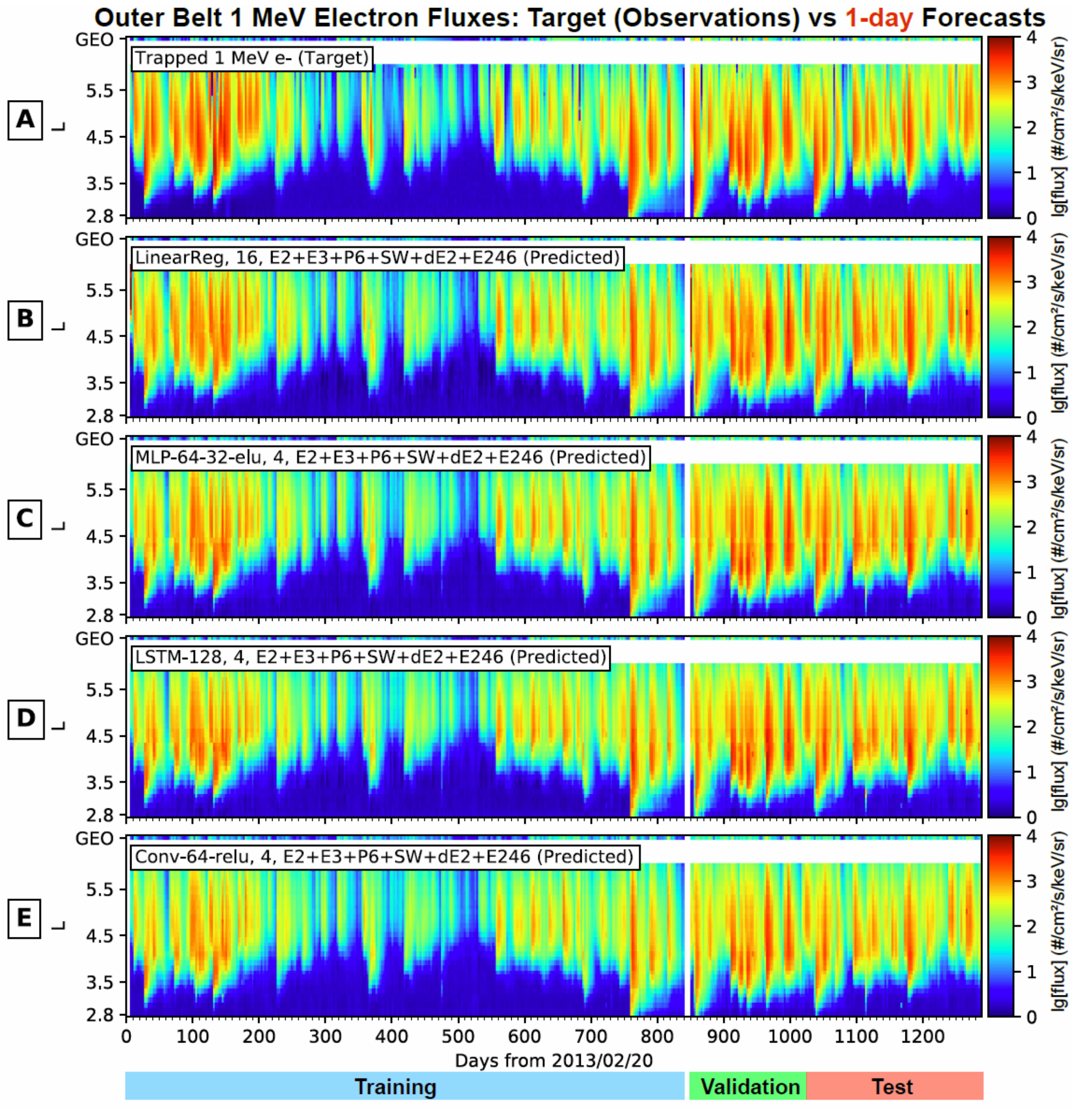


Figure 44. Overview of target and 1-day forecasted fluxes across all Lshells**.** **A** shows the observed flux distributions to be forecasted for 1 MeV electrons. **B, C, D,** and **E** show, respectively, predictions from the models with the highest PE including linear regression model, MLP, LSTM, and CNN models.

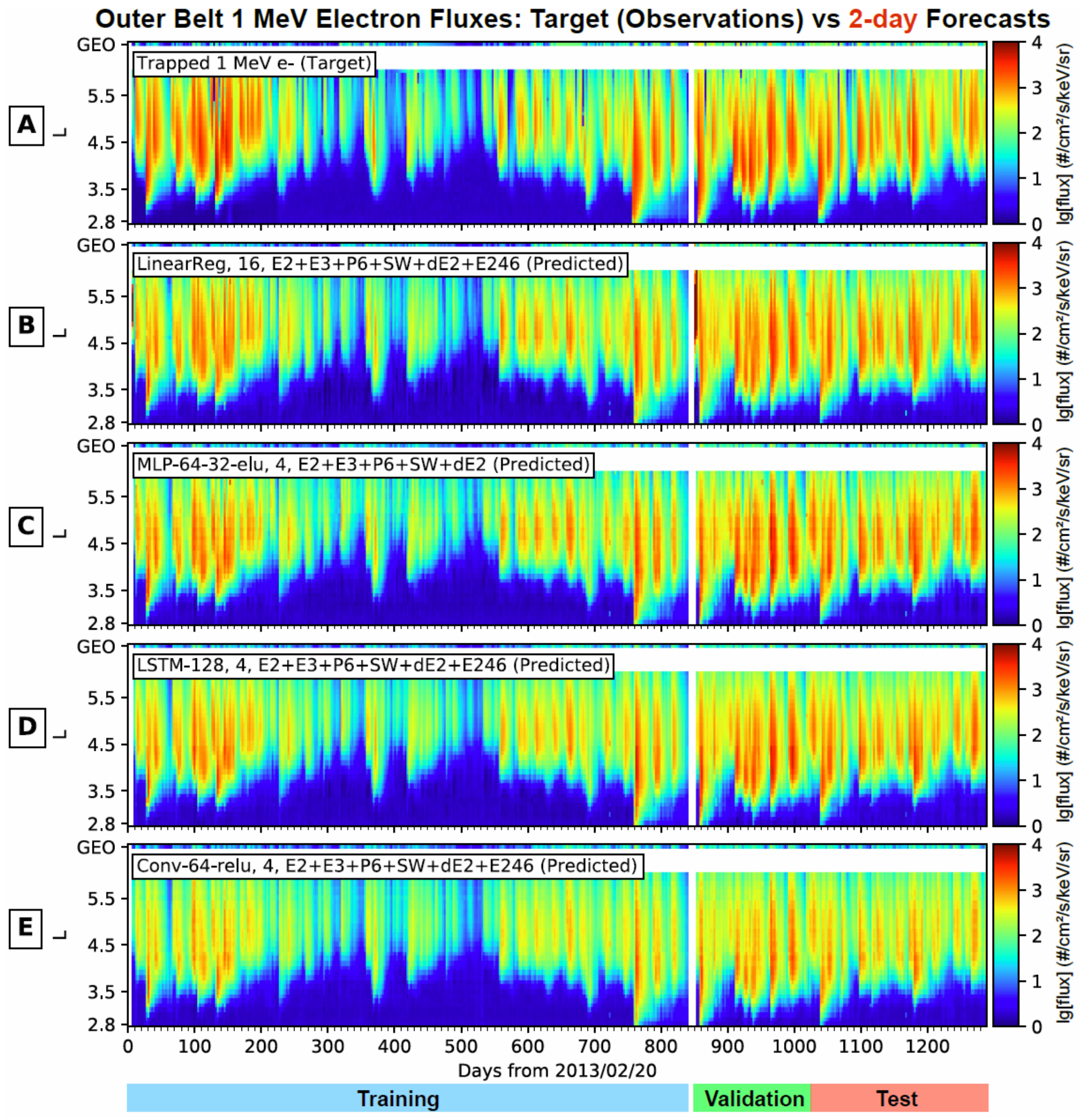


Figure 45. Overview of target and 2-day forecasted fluxes across all Lshells. All panels are in the same format as in Figure 24.

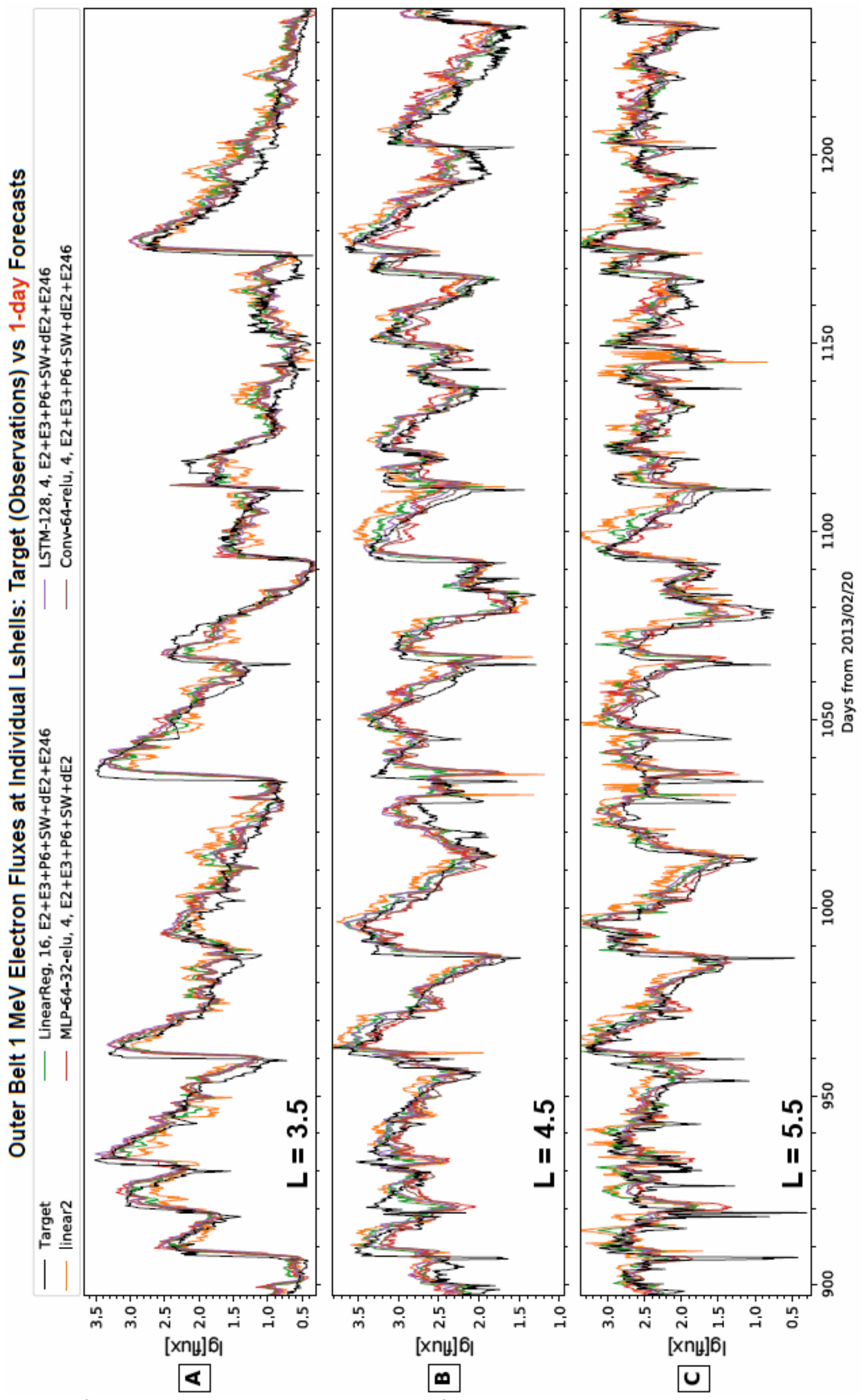


Figure 46. One-day forecasts compared to target fluxes at three selected Lshells over the combined validation and test period. A, B, and C are for Lshells of 3.5, 4.5, and 5.5, respectively. The measured 1 MeV electrons (black) are compared to predictions from the LinearReg, MLP, LSTM, and CNN models with highest PE in each category (Table 17) as well as linear2 model (yellow).

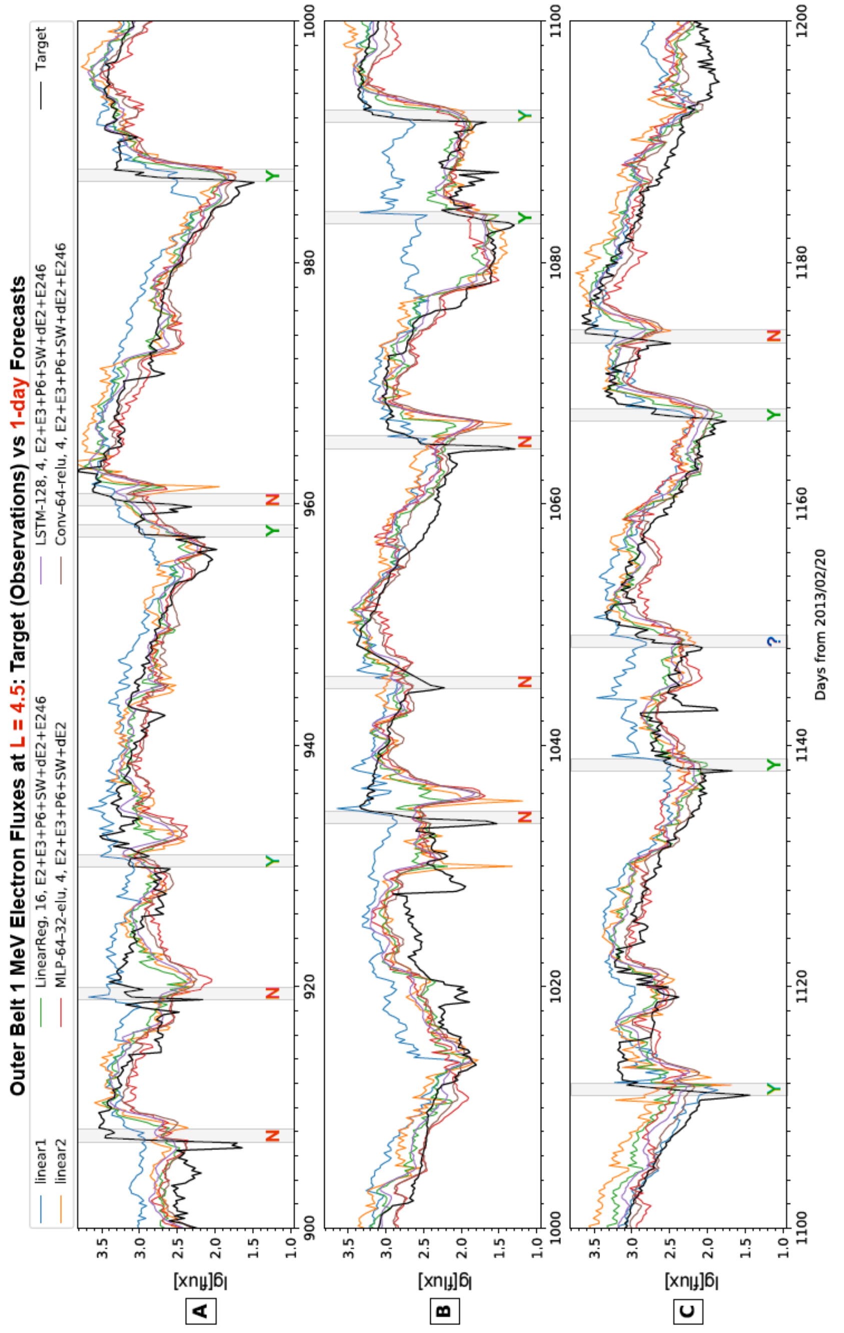


Figure 47. One-day forecasts are compared to target fluxes at one single Lshell (L=4.5) over the validation and test period. The time period is separated into three panels to show more details. Vertical gray boxes mark out 16 major MeV electron events—the left sides coincide the start of incoming MeV electron events and the width is 25 hours—and are also called prediction windows. A successful (failed, unclear) prediction of sudden MeV electron increment falls within (outside, on the edge) the prediction window and is marked with a green (red, blue) letter Y (N, ?).

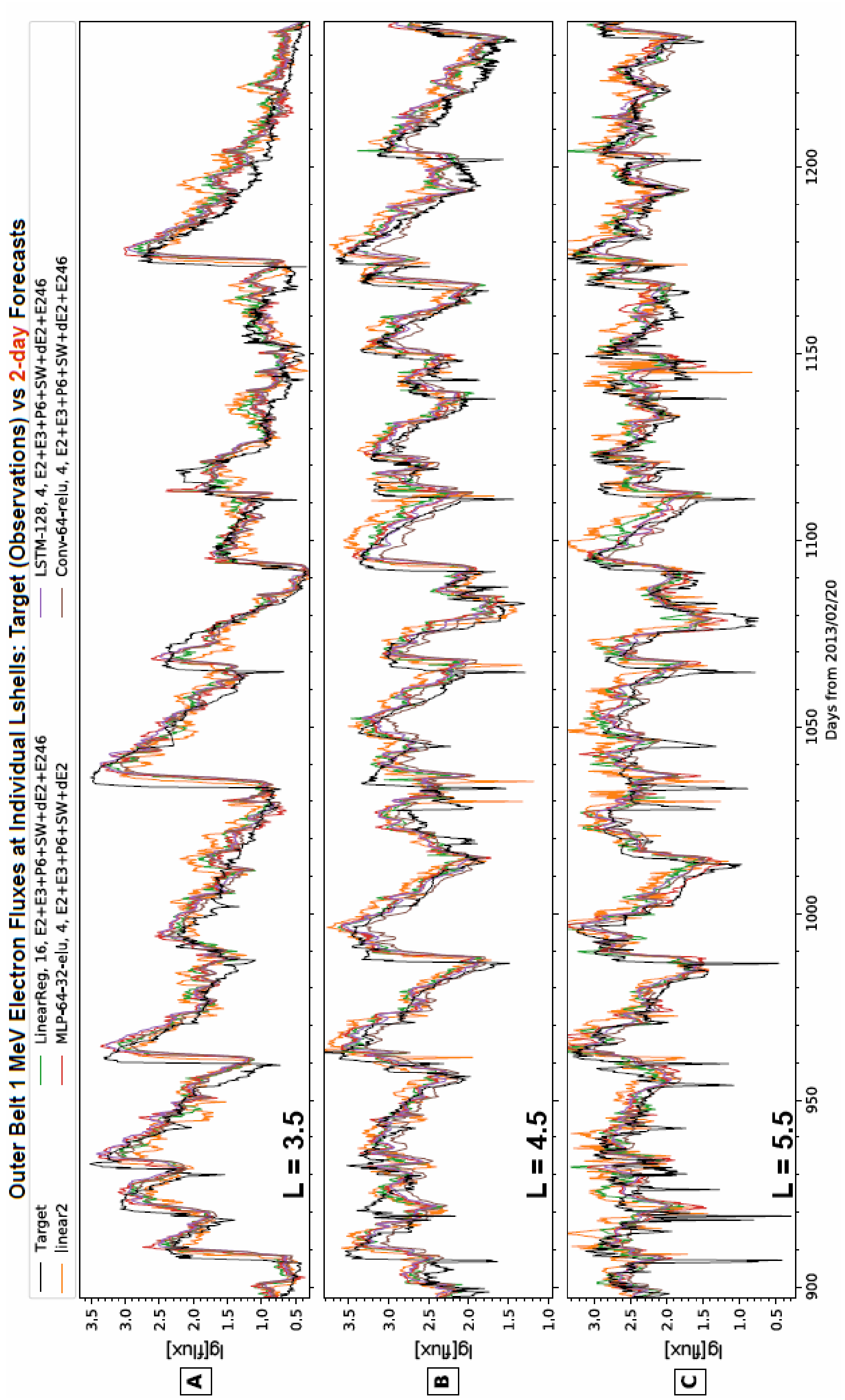


Figure 48. Two-day forecasts are compared to target fluxes at three selected Lshells over the combined validation and test period. Same format as Figure 32. Note here linear2 is for 1-day forecasts instead of 2-day.

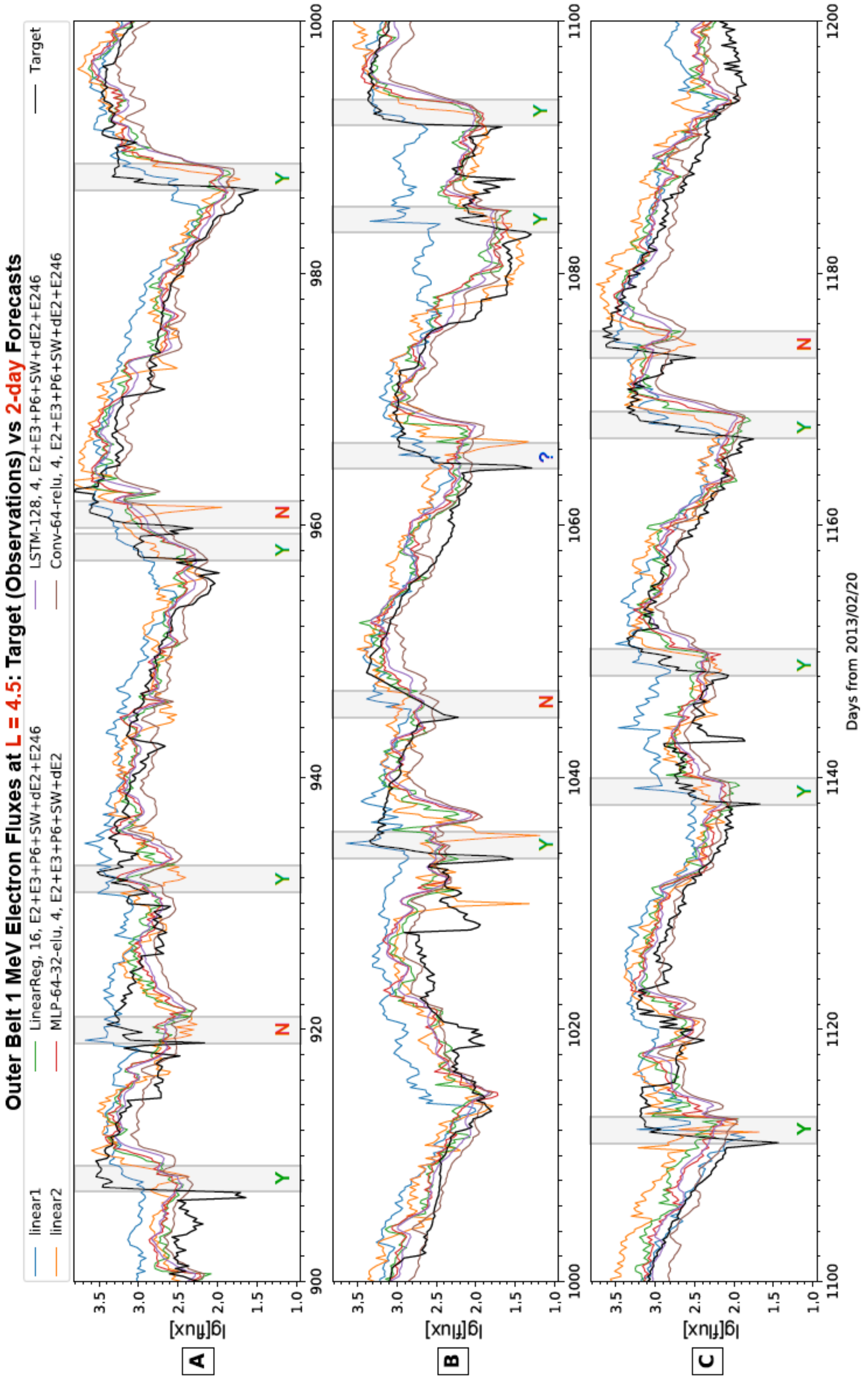


Figure 49. Two-day forecasts are compared to target fluxes at one single Lshell (L=4.5) over the combined validation and test period. Same format as Figure 33. The gray vertical boxes have a prediction window width of 50 hr. Note here linear1 and linear2 are for 1-day forecasts instead of 2-day.

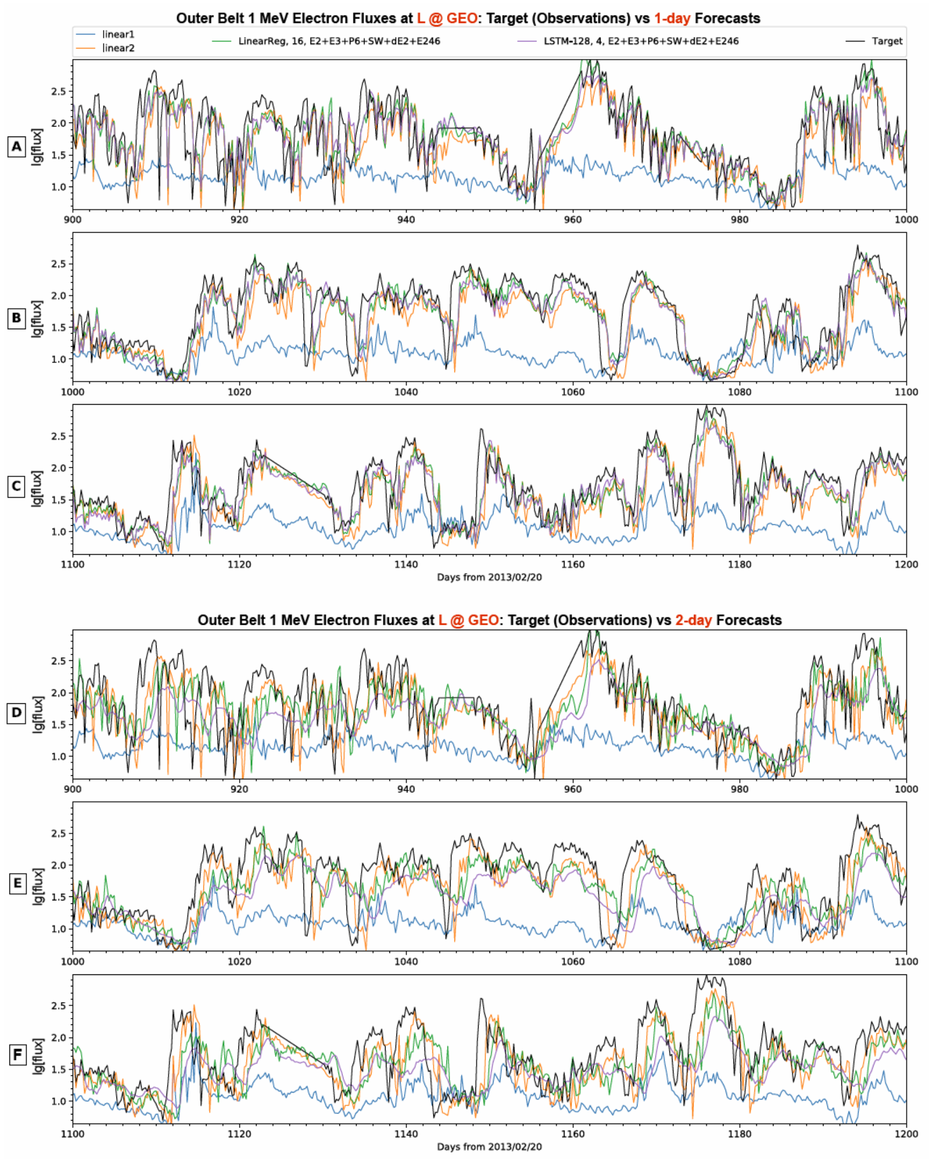


Figure 50. One-day forecasts (A, B and C) and 2-day forecasts (D, E, and F) are compared to target fluxes at GEO over the validation and test period. Same format as Figure 34. Here only results from LinearReg and LSTM models are shown for clearness. Note here linear1 and linear2 are all for 1-day forecasts.

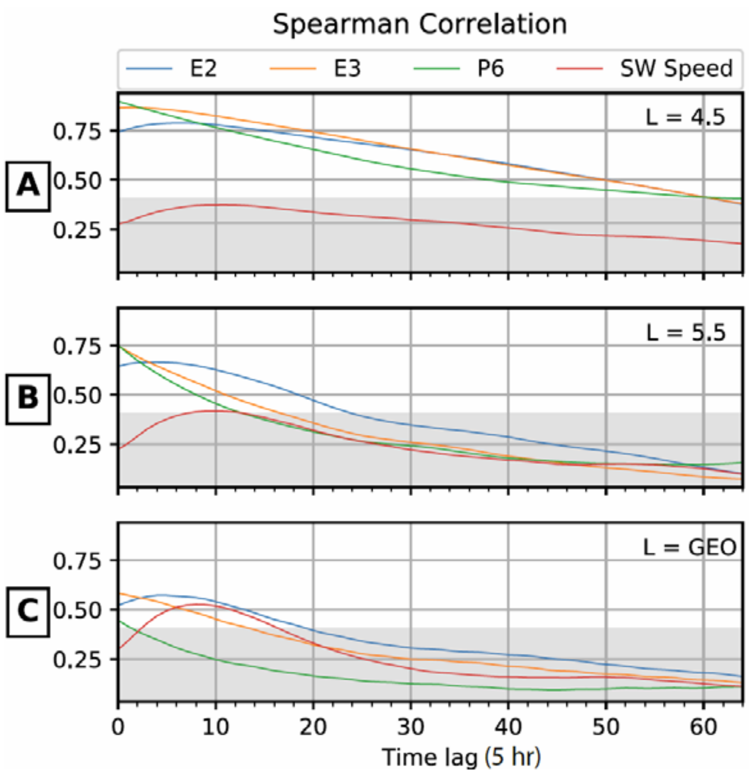


Figure 51. Spearman correlation between target and input variables for multiple L-shells. A, B, and C show, respectively, L=4.5, L=5.5, and at GEO the values of the Spearman correlation of E2, E3, P6, and the solar wind speed with the target 1 MeV electrons for different time lags. Each time lag corresponds to 5 hours. The top of each gray area corresponds to correlation value ~0.4.

Tables

Table 17. Test input parameter combinations for 1-day (25 hours) forecasts. Columns of PE values (averaged for all Lshells) are for training data, validation data, test data, validation and test data together, and all data, respectively. The last column shows PE for validation and test data at GEO only. The 10th model with the highest PE values is highlighted in red.

model	**window size**	**Input data**	**PE train**	**PE validation**	**PE test**	**PE val+test**	**PE all**	**PE GEO val+test**
linear1	15	E2/E3	0.679	0.602	0.695	0.668	0.691	-1.732
linear2	15	E2+P6	0.869	0.753	0.816	0.797	0.854	0.352
01. LinearReg	16	SW	0.569	0.289	0.634	0.518	0.574	0.557
02. LinearReg	16	E2	0.777	0.736	0.831	0.800	0.795	0.464
03. LinearReg	16	E3	0.822	0.769	0.846	0.822	0.831	0.390
04. LinearReg	16	P6	0.830	0.737	0.811	0.788	0.825	0.373
05. LinearReg	16	E2+E3	0.847	0.797	0.870	0.846	0.854	0.496
06. LinearReg	16	E2+P6	0.883	0.814	0.869	0.852	0.879	0.480
07. LinearReg	16	E2+SW	0.782	0.747	0.839	0.809	0.801	0.574
08. LinearReg	16	E2+E3+P6	0.889	0.819	0.873	0.856	0.885	0.494
09. LinearReg	16	E2+E3+P6+SW	0.896	0.830	0.874	0.861	0.890	0.584
10. LinearReg	16	E2+E3+P6+SW+dE2	**0.897**	**0.830**	**0.875**	**0.861**	**0.891**	**0.587**
11. LSTM-128	16	SW	0.503	0.052	0.551	0.382	0.490	-1.471
12. LSTM-128	16	E2	0.773	0.743	0.819	0.795	0.790	0.461
13. LSTM-128	16	E3	0.805	0.753	0.832	0.807	0.815	0.394
14. LSTM-128	16	P6	0.803	0.714	0.797	0.771	0.803	0.168
15. LSTM-128	16	E2+E3	0.852	0.768	0.851	0.824	0.850	0.468
16. LSTM-128	16	E2+P6	0.859	0.790	0.851	0.832	0.858	0.459
17. LSTM-128	16	E2+SW	0.793	0.777	0.826	0.812	0.809	0.548
18. LSTM-128	16	E2+E3+P6	0.872	0.800	0.863	0.843	0.870	0.426
19. LSTM-128	16	E2+E3+P6+SW	0.886	0.829	0.866	0.855	0.882	0.536
20. LSTM-128	16	E2+E3+P6+SW+dE2	0.884	0.826	0.865	0.853	0.880	0.564

Table 18. Performance of models in four categories for 1-day (25 hours) forecasts. Same format as Table 1. PE values for the top performer of each category are highlighted in red, also the top performers have their model index numbers marked with asterisk. E246 in the input list indicates E2 fluxes at L = 4.6.

index	model	window size	input data	PE train	PE validation	PE test	PE val+test	PE all	PE GEO val+test
	linear1	15	E2/E3	0.679	0.602	0.695	0.668	0.691	-1.732
	linear2	15	E2+P6	0.869	0.753	0.816	0.797	0.854	0.352
1	LinearReg	4	E2+E3+P6+SW	0.890	0.827	0.871	0.858	0.886	0.568
2	LinearReg	16	E2+E3+P6+SW	0.896	0.830	0.874	0.861	0.890	0.584
3	LinearReg	4	E2+E3+P6+SW+dE2	0.890	0.827	0.872	0.858	0.886	0.571
4	LinearReg	16	E2+E3+P6+SW+dE2	0.897	0.830	0.875	0.861	0.891	0.587
5	LinearReg	4	E2+E3+P6+SW+dE2+E246	0.902	0.838	0.884	0.869	0.897	0.571
*6	LinearReg	16	E2+E3+P6+SW+dE2+E246	**0.906**	**0.838**	**0.887**	**0.872**	**0.900**	**0.587**
7	MLP-64-32-elu	4	E2+E3+P6+SW	0.885	0.845	0.875	0.867	0.885	0.580
8	MLP-64-32-elu	16	E2+E3+P6+SW	0.901	0.830	0.875	0.861	0.894	0.575
9	MLP-64-32-elu	4	E2+E3+P6+SW+dE2	0.874	0.841	0.871	0.863	0.877	0.502
10	MLP-64-32-elu	16	E2+E3+P6+SW+dE2	0.888	0.834	0.871	0.860	0.885	0.363
*11	MLP-64-32-elu	4	E2+E3+P6+SW+dE2+E246	**0.899**	**0.848**	**0.879**	**0.871**	**0.895**	0.524
12	MLP-64-32-elu	16	E2+E3+P6+SW+dE2+E246	0.904	0.833	0.878	0.864	0.897	**0.590**
13	LSTM-128	4	E2+E3+P6+SW	0.877	0.837	0.874	0.863	0.879	0.448
14	LSTM-128	16	E2+E3+P6+SW	0.884	0.822	0.862	0.850	0.879	0.530
15	LSTM-128	4	E2+E3+P6+SW+dE2	0.877	0.841	0.879	0.868	0.880	0.444
16	LSTM-128	16	E2+E3+P6+SW+dE2	0.886	0.830	0.870	0.859	0.883	0.558
*17	LSTM-128	4	E2+E3+P6+SW+dE2+E246	**0.893**	**0.846**	**0.886**	**0.874**	**0.892**	**0.589**
18	LSTM-128	16	E2+E3+P6+SW+dE2+E246	0.899	0.830	0.872	0.859	0.892	0.536
19	Conv-64-relu	4	E2+E3+P6+SW	0.872	0.835	0.878	0.865	0.876	0.363
20	Conv-64-32-relu	16	E2+E3+P6+SW	0.799	0.696	0.791	0.761	0.797	-0.121
21	Conv-64-relu	4	E2+E3+P6+SW+dE2	0.872	0.838	0.884	0.870	0.877	0.352
22	Conv-64-32-relu	16	E2+E3+P6+SW+dE2	0.787	0.664	0.776	0.740	0.783	-0.112
*23	Conv-64-relu	4	E2+E3+P6+SW+dE2+E246	**0.891**	**0.847**	**0.890**	**0.877**	**0.892**	**0.363**
24	Conv-64-32-relu	16	E2+E3+P6+SW+dE2+E246	0.796	0.677	0.770	0.741	0.789	-0.308

Table 19. Performance of models in four categories for 2-day (50 hours) forecasts. Same format as Table 2. Note the PE values for linear 1 and linear2 are for 1-day forecasts instead of 2-day.

index	model	window size	input data	PE train	PE validation	PE test	PE val+test	PE all	PE GEO val+test
	linear1	15	E2/E3	0.679	0.600	0.695	0.667	0.691	-1.723
	linear2	15	E2+P6	0.869	0.752	0.816	0.797	0.854	0.352
1	LinearReg	4	E2+E3+P6+SW	0.853	0.772	0.837	0.817	0.849	0.269
2	LinearReg	16	E2+E3+P6+SW	0.858	0.774	0.841	0.820	0.854	0.323
3	LinearReg	4	E2+E3+P6+SW+dE2	0.853	0.772	0.838	0.818	0.850	0.282
4	LinearReg	16	E2+E3+P6+SW+dE2	0.860	0.773	0.842	0.820	0.855	0.333
5	LinearReg	4	E2+E3+P6+SW+dE2+E246	0.865	0.779	0.844	0.824	0.859	0.282
*6	LinearReg	16	E2+E3+P6+SW+dE2+E246	**0.871**	**0.778**	**0.849**	**0.827**	**0.864**	**0.333**
7	MLP-64-32-elu	4	E2+E3+P6+SW	0.846	0.775	0.838	0.819	0.845	0.102
8	MLP-64-32-elu	16	E2+E3+P6+SW	0.859	0.765	0.829	0.809	0.851	0.285
*9	MLP-64-32-elu	4	E2+E3+P6+SW+dE2	**0.857**	**0.780**	**0.842**	**0.823**	**0.854**	0.200
10	MLP-64-32-elu	16	E2+E3+P6+SW+dE2	0.863	0.772	0.837	0.817	0.855	0.326
11	MLP-64-32-elu	4	E2+E3+P6+SW+dE2+E246	0.860	0.776	0.835	0.817	0.854	-0.037
12	MLP-64-32-elu	16	E2+E3+P6+SW+dE2+E246	0.859	0.772	0.832	0.814	0.852	**0.353**
13	LSTM-128	4	E2+E3+P6+SW	0.835	0.760	0.833	0.810	0.835	0.073
14	LSTM-128	16	E2+E3+P6+SW	0.847	0.757	0.825	0.804	0.842	0.208
15	LSTM-128	4	E2+E3+P6+SW+dE2	0.840	0.767	0.837	0.815	0.840	0.119
16	LSTM-128	16	E2+E3+P6+SW+dE2	0.851	0.762	0.831	0.810	0.846	0.307
*17	LSTM-128	4	E2+E3+P6+SW+dE2+E246	**0.853**	**0.770**	**0.839**	**0.818**	**0.849**	0.104
18	LSTM-128	16	E2+E3+P6+SW+dE2+E246	0.865	0.765	0.830	0.811	0.855	**0.353**
19	Conv-64-relu	4	E2+E3+P6+SW	0.841	0.771	0.840	0.819	0.842	0.080
20	Conv-64-32-relu	16	E2+E3+P6+SW	0.765	0.607	0.742	0.699	0.756	-0.498
21	Conv-64-relu	4	E2+E3+P6+SW+dE2	0.840	0.761	0.839	0.814	0.840	0.056
22	Conv-64-32-relu	16	E2+E3+P6+SW+dE2	0.769	0.596	0.731	0.688	0.755	-0.512
*23	Conv-64-relu	4	E2+E3+P6+SW+dE2+E246	**0.846**	**0.767**	**0.844**	**0.820**	**0.845**	-0.073
24	Conv-64-32-relu	16	E2+E3+P6+SW+dE2+E246	0.771	0.583	0.715	0.673	0.753	-0.636

Acknowledgments, Samples, and Data

The authors declare no conflicts of interest. Pires de Lima acknowledges CNPq (grant no. 203589/2014-9) for graduate sponsorship. We gratefully acknowledge the support of NASA Heliophysics Space Weather Operations to Research Program (18-HSWO2R18-0006), the NASA Heliophysics Guest Investigators program (14-GIVABR14_2-0028), and LANL internal funding. We want to acknowledge the PIs and instrument teams of NOAA POES SEM2, and RBSP EMFISIS for providing measurements and allowing us to use their data. Thanks to CDAWeb for providing OMNI data. RBSP and POES data used in this work were downloadable from the missions' public websites (https://www.rbsp-ect.lanl.gov, and http://www.ngdc.noaa.gov), while LANL GEO data are available from AGU website as supplementary material of Chen et al. (2019).

References

Ayodele, T.O., 2010. Types of machine learning algorithms, in: New Advances in Machine Learning. IntechOpen.

Belian, R.D., Gisler, G.R., Cayton, T., Christensen, R., 1992. High- Z energetic particles at geosynchronous orbit during the Great Solar Proton Event Series of October 1989. J. Geophys. Res. 97, 16897. https://doi.org/10.1029/92JA01139

Blake, J.B., Carranza, P.A., Claudepierre, S.G., Clemmons, J.H., Crain, W.R., Dotan, Y., Fennell, J.F., Fuentes, F.H., Galvan, R.M., George, J.S., Henderson, M.G., Lalic, M., Lin, A.Y., Looper, M.D., Mabry, D.J., Mazur, J.E., McCarthy, B., Nguyen, C.Q., O'Brien, T.P., Perez, M.A., Redding, M.T., Roeder, J.L., Salvaggio, D.J., Sorensen, G.A., Spence, H.E., Yi, S., Zakrzewski, M.P., 2013. The Magnetic Electron Ion Spectrometer (MagEIS) Instruments Aboard the Radiation Belt Storm Probes (RBSP) Spacecraft. Space Sci. Rev. 179, 383–421. https://doi.org/10.1007/s11214-013-9991-8

Camporeale, E., 2019. The Challenge of Machine Learning in Space Weather: Nowcasting and Forecasting. Sp. Weather 2018SW002061. https://doi.org/10.1029/2018SW002061

Chen, Y., Reeves, G.D., Fu, X., Henderson, M., 2019. PreMevE: New Predictive Model for Megaelectron-Volt Electrons Inside Earth's Outer Radiation Belt. Sp. Weather 17, 438–454. https://doi.org/10.1029/2018SW002095

Clevert, D.-A., Unterthiner, T., Hochreiter, S., 2015. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). arXiv e-prints arXiv:1511.07289.

Cracknell, M.J., Reading, A.M., 2014. Geological mapping using remote sensing data: A comparison of five machine learning algorithms, their response to variations in the spatial distribution of training data and the use of explicit spatial information. Comput. Geosci. 63, 22–33. https://doi.org/10.1016/J.CAGEO.2013.10.008

Duarte-Coronado, D., Tellez-Rodriguez, J., Pires de Lima, R., Marfurt, K., Slatt, R., 2019. Deep convolutional neural networks as an estimator of porosity in thin-section images for unconventional reservoirs, in: SEG Technical Program Expanded Abstracts 2019. Society of Exploration Geophysicists, pp. 3181–3184. https://doi.org/10.1190/segam2019-3216898.1

Dumoulin, V., Visin, F., 2016. A guide to convolution arithmetic for deep learning. ArXiv e-prints.

Evans, D.S., Greer, M.S., (U.S.), S.E.C., 2000. Polar orbiting environmental satellite space environment monitor-2: instrument description and archive data documentation. U.S. Dept. of Commerce, National Oceanic and Atmospheric Administration, Oceanic and Atmospheric Research Laboratories, Space Environment Center, Boulder, CO.

Graves, A., Jaitly, N., Mohamed, A., 2013. Hybrid speech recognition with Deep Bidirectional LSTM, in: 2013 IEEE Workshop on Automatic Speech Recognition and Understanding. IEEE, pp. 273–278. https://doi.org/10.1109/ASRU.2013.6707742

Graves, A., Schmidhuber, J., 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks 18, 602–610. https://doi.org/10.1016/J.NEUNET.2005.06.042

Hahnloser, R.H.R., Sarpeshkar, R., Mahowald, M.A., Douglas, R.J., Seung, H.S., 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature 405, 947–951. https://doi.org/10.1038/35016072

Hartigan, J.A., Wong, M.A., 1979. Algorithm AS 136: A K-Means Clustering Algorithm. J. R. Stat. Soc. Ser. C (Applied Stat. 28, 100–108. https://doi.org/10.2307/2346830

Hochreiter, S., Schmidhuber, J., 1997. Long Short-Term Memory. Neural Comput. 9, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. U. S. A. 79, 2554–8. https://doi.org/10.1073/pnas.79.8.2554

Horne, R.B., Glauert, S.A., Meredith, N.P., Koskinen, H., Vainio, R., Afanasiev, A., Ganushkina, N.Y., Amariutei, O.A., Boscher, D., Sicard, A., Maget, V., Poedts, S., Jacobs, C., Sanahuja, B., Aran, A., Heynderickx, D., Pitchford, D., 2013. Forecasting the Earth’s radiation belts and modelling solar energetic particle events: Recent results from SPACECAST. J. Sp. Weather Sp. Clim. 3, A20. https://doi.org/10.1051/swsc/2013042

Kingma, D.P., Ba, J., 2014. Adam: A Method for Stochastic Optimization. arXiv e-prints arXiv:1412.6980.

Kitamura, K., Nakamura, Y., Tokumitsu, M., Ishida, Y., Watari, S., 2011. Prediction of the electron flux environment in geosynchronous orbit using a neural network technique. Artif. Life Robot. 16, 389–392. https://doi.org/10.1007/s10015-011-0957-1

Kong, W., Dong, Z.Y., Jia, Y., Hill, D.J., Xu, Y., Zhang, Y., 2019. Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. IEEE Trans. Smart Grid 10, 841–851. https://doi.org/10.1109/TSG.2017.2753802

Kortström, J., Uski, M., Tiira, T., 2016. Automatic classification of seismic events within a regional seismograph network. Comput. Geosci. 87, 22–30. https://doi.org/10.1016/J.CAGEO.2015.11.006

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet Classification with Deep Convolutional Neural Networks, in: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS’12. Curran Associates Inc., USA, pp. 1097–1105.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444. https://doi.org/10.1038/nature14539

Mauk, B.H., Fox, N.J., Kanekal, S.G., Kessel, R.L., Sibeck, D.G., Ukhorskiy, A., 2013. Science Objectives and Rationale for the Radiation Belt Storm Probes Mission. Space Sci. Rev. 179, 3–27. https://doi.org/10.1007/s11214-012-9908-y

McIlwain, C.E., 1966. Magnetic coordinates. Space Sci. Rev. 5, 585–598. https://doi.org/10.1007/BF00167327

Minsky, M., 1961. Steps toward Artificial Intelligence. Proc. IRE 49, 8–30. https://doi.org/10.1109/JRPROC.1961.287775

Murdoch, W.J., Singh, C., Kumbier, K., Abbasi-Asl, R., Yu, B., 2019. Definitions, methods, and applications in interpretable machine learning. Proc. Natl. Acad. Sci. 116, 22071 LP – 22080. https://doi.org/10.1073/pnas.1900654116

Nair, V., Hinton, G.E., 2010. Rectified Linear Units Improve Restricted Boltzmann Machines, in: Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML’10. Omnipress, USA, pp. 807–814.

Olah, C., 2015. Understanding LSTM Networks.

Olson, W.P., Pfitzer, K.A., 1977. Magnetospheric magnetic field modeling. Annual scientific report. Huntington Beach.

Perol, T., Gharbi, M., Denolle, M., 2018. Convolutional neural network for earthquake detection and location. Sci. Adv. 4, e1700578. https://doi.org/10.1126/sciadv.1700578

Pires de Lima, R., Bonar, A., Coronado, D.D., Marfurt, K., Nicholson, C., 2019a. Deep convolutional neural networks as a geological image classification tool. Sediment. Rec. 17, 4–9. https://doi.org/10.210/sedred.2019.2

Pires de Lima, R., Marfurt, K.J., 2018. Principal component analysis and K-means analysis of airborne gamma-ray spectrometry surveys, in: SEG Technical Program Expanded Abstracts 2018. Society of Exploration Geophysicists, pp. 2277–2281. https://doi.org/10.1190/segam2018-2996506.1

Pires de Lima, R., Suriamin, F., Marfurt, K.J., Pranter, M.J., 2019b. Convolutional neural networks as aid in core lithofacies classification. Interpretation 7, SF27–SF40. https://doi.org/10.1190/INT-2018-0245.1

Qayyum, A., Anwar, S.M., Awais, M., Majid, M., 2017. Medical image retrieval using deep convolutional neural network. Neurocomputing 266, 8–20. https://doi.org/10.1016/J.NEUCOM.2017.05.025

Ren, C.X., Dorostkar, O., Rouet-Leduc, B., Hulbert, C., Strebel, D., Guyer, R.A., Johnson, P.A., Carmeliet, J., 2019. Machine Learning Reveals the State of Intermittent Frictional Dynamics in a Sheared Granular Fault. Geophys. Res. Lett. 46, 7395–7403. https://doi.org/10.1029/2019GL082706

Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation, in: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (Eds.), Medical Image Computing and Computer-Assisted Intervention -- MICCAI 2015. Springer International Publishing, Cham, pp. 234–241.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vis. 115, 211–252. https://doi.org/10.1007/s11263-015-0816-y

Shin, D.-K., Lee, D.-Y., Kim, K.-C., Hwang, J., Kim, J., 2016. Artificial neural network prediction model for geosynchronous electron fluxes: Dependence on satellite position and particle energy. Sp. Weather 14, 313–321. https://doi.org/10.1002/2015SW001359

Sinha, S., Wen, Y., Pires de Lima, R.A., Marfurt, K., 2018. Statistical controls on induced seismicity. Unconventional Resources Technology Conference. https://doi.org/10.15530/urtec-2018-2897507-MS

Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M., 2014. Striving for Simplicity: The All Convolutional Net. arXiv e-prints arXiv:1412.6806.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J. Mach. Learn. Res. 15, 1929–1958.

Stringer, G.A., Heuten, I., Salazar, C., Stokes, B., 1996. Artificial Neural Network (ANN) Forecasting of Energetic Electrons at Geosynchronous Orbit. American Geophysical Union (AGU), pp. 291–295. https://doi.org/10.1029/GM097p0291

Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B., Liang, J., 2016. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? IEEE Trans. Med. Imaging 35, 1299–1312. https://doi.org/10.1109/TMI.2016.2535302

Ukhorskiy, A.Y., Sitnov, M.I., Sharma, A.S., Anderson, B.J., Ohtani, S., Lui, A.T.Y., 2004. Data-derived forecasting model for relativistic electron intensity at geosynchronous orbit. Geophys. Res. Lett. 31, L09806. https://doi.org/10.1029/2004GL019616

Valentín, M.B., Bom, C.R., Coelho, J.M., Correia, M.D., de Albuquerque, Márcio P., de Albuquerque, Marcelo P., Faria, E.L., 2019. A deep residual convolutional neural network for automatic lithological facies identification of Brazilian pre-salt oilfield wellbore image logs. J. Pet. Sci. Eng. https://doi.org/10.1016/J.PETROL.2019.04.030

Wang, T., Zhang, Z., Li, Y., 2019. EarthquakeGen: Earthquake generator using generative adversarial networks, in: SEG Technical Program Expanded Abstracts 2019. Society of Exploration Geophysicists, pp. 2674–2678. https://doi.org/10.1190/segam2019-3216687.1

Wei, L., Zhong, Q., Lin, R., Wang, J., Liu, S., Cao, Y., 2018. Quantitative Prediction of High-Energy Electron Integral Flux at Geostationary Orbit Based on Deep Learning. Sp. Weather 16, 903–916. https://doi.org/10.1029/2018SW001829

Conclusions and final remarks

I started this dissertation with a chapter that explains the more important concepts of machine learning, in general, adding greater details on the internal components of convolutional neural networks. The following chapters showed different applications of convolutional neural networks ranging from core analysis, fossil identification, and remote sensing analysis, many times relying on transfer learning, to exploit tools that can help geoscientists working on different fields. Despite my evident predilection to convolutional neural networks, the results in chapter 6 show that simpler models should not be overlooked when a new analysis starts. Simpler models, including linear regression, are often easier to explain and should serve as a starting point in many different tasks.

This dissertation heavily focused on applications of classification with supervised learning, with some applications of regression in the final chapter. I briefly mention other machine learning types in chapter 1 that are likely more appropriate for 3D seismic volumes, such as convolutional neural networks used for segmentation tasks. As such models have a higher dimensional output, e.g. a 2D plane or 3D volume instead of a single scalar, and are the basis for many applications of machine learning for seismic facies classification and seismic inversion. Also briefly mentioned in the first chapter, reinforcement learning is one of the machine learning types producing some of the most curious results, such as beating professional players in games highly dependent on strategy. Although I have spoken with researchers evaluating the use of reinforcement learning techniques on geoscience problems, I could not find any references on the subject. There are many research areas to be explored.