MNIST Image Classification with Convolutional Neural Network

Jason Rich
Old Dominion University
Computer Science
Norfolk, Virginia 23504
jrich069@odu.edu

***Abstract*—In this article, I will demonstrate the use of a Convolutional Neural Network (CNN) as a technique for image classification. The dataset used for this study is The MNIST database of handwritten digits, which contains a training set of 60,000 examples, and a test set of 10,000 examples. The dataset is a subset of the larger set available from National Institute of Standards and Technology LeCun et al. (1998). The goal of this paper is to show that analyzing the MNIST data, using Anaconda's python 3.5 distribution, and Google's TensorFlow package for python3, on a standard laptop is not only possible, but also efficient, accurate, and certainly affordable. Moreover, I will show that CNN will converge in as little as 2000 steps, and that as the steps increase, the error rate draws closer and closer to zero, as the accuracy of the model grows closer and closer to 100%.**

I. Introduction

Historically, to perform image processing, whether high quality digital examples, or hand written notes presented using a standard office scanner, the machine learning practitioner would have to extract language dependent features like curvature of different letters, spacing, black and white letters, etc., only to use a classifier such as Support Vector Machine (SVM) to distingish between writers Dwivedi (2018). With the publication of (LeCun et al. 1998), the analysis of handwritten, variable, 2D shapes with Convolutional Neural Network was shown to outperform all other techniques LeCun et al. (1998).

I will show that given the advances in Application Program Interface (API) frameworks, such as TensorFlow TensorFlow (n.d.), Keras keras-team (n.d.), and H2O Ambati and Click (n.d.), have provided machine learning researchers and practitioners the ability and tools to quickly and efficiently analyze larges amounts of data. The latest API releases reduce the cost of what are traditionally thought of as mathematically complex and computationally expensive algorithms, making computation less complex and affordable.

The key observation in this study is, given a well studied dataset and an evolving deep learning algorithm, the ability of personal hardware, in my case my 2011 Mac Book Pro, with 16GB of RAM, a 1TB harddrive, and an i5 Intel processor, to reproduce results originally calculated on academic or remote research servers. This says a lot about the hardware, but more so about the work, research, and improvements that have rolled-up into the current versions of modern day deep learning algorithms.

Hopefully, by the conclusion of this paper, I will have shown, that we have come a long way the field of deep learning. However, I also hope to show that we have much more work remaining, and efforts in the fields of quantum machine learning, quantum deep learning, and continued improvment in high performance computing, are quintessential to further the advancements, demonstrated within this paper.

II. Related Work

A. Foundational Work

LeCun et al. (1998) laid the foundationalground work for all current convolutional neural network architecture and image processing, building on the concepts of Gradient-Based Learning. The work of LeCun et al. (1998), and others, set the tone for work that is happening today. Without the work of people like LeCun, Hinton, and Ng, we may not have the bleeding edge algorithms or the tools to analyze the data we can today.

B. Gradient-Based Learning

The general problem of minimizing a function with respect to a set of parameter is at the root of many issues in computer science. Gradient-Based Learning draws on the fact that it is generally much easier to minimize a reasonably smooth, continuous function than a discrete (combinatorial) function. This is measured by the gradient of the loss function with repect to the parameters. Efficient learning algorithms can be devised when the gradient vector can be computed analytically (as opposed to numerically through perturbation). Furthermore, LeCun et al. (1998) notes; ...the basis of numerous gradient-based learning algorithms with continuous-valued parameters. In the procedure described continuous-values parameters W is a real-valued vector, with respect to which $E(W)$ is continuous, as well as differentiable almost everywhere. [T]he simplest minimization procedure in such a setting is the gradient descent algorithm where W is iteratively adjusted as follows:

$$W_k = W_{k-1} - \epsilon \frac{\partial \mathbf{E}(W)}{\partial W} \tag{1}$$

In the simplest case, ϵ is a scalor constant LeCun et al. (1998). Moreover, LeCun et al. (1998) note: A poplar minimization procedure is the stochastic gradient algorithm,

also called the the on-line update. It consists in updating the parameter vector using a noisy, or approximated, version of the average gradient. In the most common instance of it, W is updated on the basis of a single sample:

$$W_k = W_{k-1} - \epsilon \frac{\partial \mathbf{E}^{p_k}(W)}{\partial W} \tag{2}$$

With this procedure the parameter vector fluctuates around an average trajectory, but usually converges considerably faster than a regular gradient descent and second order methods on large training set with redundant sample. . . LeCun et al. (1998). For more information on stochastic gradient descent models see Bottou (2010) and Sutskever et al. (2013).

C. Image Processing

However, with the advent of more sophisticated digital carmeras, with great pixel quality and pixels per-inch, images become larger and larger. The traditional methods of image classification, using a fully-connected network with hundreds of hidden units in the first layer LeCun et al. (1998), Bishop (2006), Goodfellow, Bengio, and Courville (2016), creates thousands of weights. Furthermore, using a fully-connected network negates the fact that neighboring pixels are more correlated that non-neighboring pixels Bishop (2006).

The primary advantage of using a convolutional neural network is the convolution itself. Convolutional neural networks are specifically designed for processing data that has a known grid-like topology Goodfellow, Bengio, and Courville (2016). Image data, as noted in Goodfellow, Bengio, and Courville (2016), should be thought of as a 2-D grid of pixels. I will provide a brief summary of convolution in section III, as well as the key differences in machine learning and deep learning.

III. Convolutional Neural Net

A. Convolution: The importance of convolution cannot be overstated. So what is convolution? Goodfellow, Bengio, and Courville (2016) describe convolution as a mathematical operation, which is specialized kind of linear operation. In its simplest form, a convolution is an operation on two functions of a real-value argument Goodfellow, Bengio, and Courville (2016).

The functions of a covolution: $x(t)$ which the output function at time t, and x and t are real-valued, and could provide different values at different points in time. $w(a)$ is the weighting function at age a, and provides a weighted average, applying more weight to recent measurements, and less weight to older measurements. When we apply the $w(a)$ function at every moment, we obtain a new function S, providing a smoothed estimate at every instance $w(t)$:

$$S(t) = \int x(a)w(t-a)da \tag{3}$$

In convolutional neural network terminology, $x(t)$ is the input, and $x(a)$ is the kernal. The output of the convolutional network is sometimes referred to as the feature map Goodfellow, Bengio, and Courville (2016), annotated another way:

$$S(t) = (x * w)(t) \tag{4}$$

w must be a valid probability density function, or the out will not be a weighted average. However, this is not usually the case Goodfellow, Bengio, and Courville (2016). At its core, convolution is defined for any function for which $S(t) = \int x(a)w(t-a)da$ is defined. The MNIST data used in this study is constructed of discrete data in the response feature (the image labels), which alters our approach, only slightly, to account for the discrete data structures Goodfellow, Bengio, and Courville (2016). It is worth noting that the major difference in the continuous form and the discrete form is: for discrete data structure, $x(t)$ can only take on interger values LeCun et al. (1998) and Goodfellow, Bengio, and Courville (2016).

Assuming the x and w are defined only on integer t, the discrete convolution is defined as:

$$S(t) = (x * w)(t) = \sum_{a=\neg\infty}^{\infty} x(a)w(t-a) \tag{5}$$

As noted in section I., convolutional networks provided machine learning practitioners the ability to analyze large datasets, with complex data structures, on what amounts to commodity hardware. The last point I want to make regarding convolution is the construct that assists convolution in making the aforementioned true. Convolution neural networks, by-way of the mathematical properties of convolution, employ a sparse interactions connectivity model Goodfellow, Bengio, and Courville (2016). Unlike typical neural networks that employ a full contected architecture, convolutional neural networks use a smaller kernal, thus require fewer connection within the network between layers. Futhermore, CNNs require the storage of fewer parameters, have a reduced memory requirement, statistical efficiency is improved, and computing the output requires fewer operations LeCun et al. (1998), Bishop (2006), and Goodfellow, Bengio, and Courville (2016). A comparison of the time complexity tells the full story. In a traditional neural network, with M inputs, and m outputs, matrix multiplication requires mxn parameters, with time complexity of $O(m$x$n)$. Conversely, convolutional neural networks limit the number of connections to k, where $m > k$, reducing the required number of parameters to kxn, and a reduced time complexity to $O(k$x$n)$, thus leading to faster and more efficient runtimes.

B. Deep Learning: Deep learning is a subset of machine learning, and refers to the depth of the layers within the neural network, and not necessarily the depth of the knowledge or wisdom obtained from utilizing the algorithms. Deep learning employs the philosophy of building complex concepts from simple concepts Goodfellow, Bengio, and Courville (2016). Take for example, image processing: The image of a persons face, a busy city street, or an excited dog; all contain a high level of complexity within the data structure that is the pixels

of the image. What deep learning provides is a mechanism for simplifying the complexity of the images pixelated data structure into smaller, more manageable chunks for processing and analysis.

Deep learning, however, is more than just convolutional neural networks. Other deep learning algorithms include: miltilayer perceptron (MLP), autoencoders, recurrent neural networks, recursive neural networks, as well as others. While I am only discussing convolutional neural networks in this paper, the reader should see Goodfellow, Bengio, and Courville (2016) for an indepth study of deep learning.

IV. Experiment

A. Dataset

The dataset used for the study is derived from The National Institute of Standards and Technology (NIST) study, and is called MNIST, where "M" stands for "modified" TensorFlow (n.d.), Goodfellow, Bengio, and Courville (2016), extracted using TensorFlow. The dataset used for this study is a subset of a much larger dataset, orignally made available by NIST LeCun et al. (1998). It consist of 60,000 images for training the models, and 10,000 images for testing the models.

The images in the dataset were pre-processed and stored as a greyscale, centered $28x28$ fixed-size image. The pre-processing performed on the images greatly improves the algorithms ability to process the data, thus assisting in minimizing the error rate.

Other than the image files, the dataset also includes the label for classifying the images. The values of the labels are on a range from 0 to 9. The image training dataset is approximately 0.099 gigabytes and the image testing dataset is considerably smaller.

The dataset was pulled locally using the `tensorflow.examples.tutorials.mnist` module, and calling `input_data` function with one hot encoding. I will fully explain the code in the next subsection, and the code is available one directory back, or on my `GitHub` account `https://github.com/jrich8573/MNIST-CNN`

B. Code

As stated above, the code used in the study was python3 along with TensorFlow open-source python3 module. I wrapped many of TensorFlow's built in fuctions with user-defined helper functions, in order to insert a great level of control over the behavior and data manipulation, otherwise left to python and TensorFlow to handle. The code was written using Anaconda's python 3 distribution (version 3.5) Anaconda (n.d.), within the Spider IDE, and a user-defined virtual environment. The code does require python $>$ 3.5. I performed the training on four separate scenarios, steps at 1000, 2500, 5000, and 10000 all benchmarked from zero time, at the starting point, while analyzing the same training data.

Obvisously, my code is not perfect, and I have certain ideas about improvemens such as including code for parallel computing, GPU, or quantum machine learning architecture (we are still years from true quantum machine learning). However,

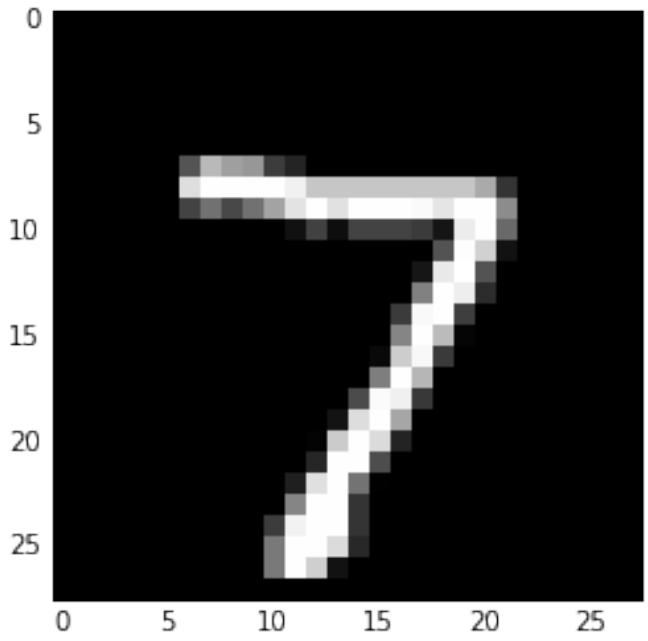

Fig. 1. Image 1

I am very happy with the results, the summary (provide in the following subsection), and the steps to convergence. However, as stated in the subsequent subsection, the code is available on my `GitHub` page, where recommendations and contributions are always a welcomed event. To contribute to the repo, please use the following workflow. First, fork repo to your account using the fork option embedded within `GitHub`. Secondly, follow the workflow below:

```
mkdir working
cd working
git clone https://github.com/$USER/MNIST-CNN.git
git checkout -b $BRANCH-NAME
```

After completing your changes and pushing them to you folked repo, please submit a pull request by using either the `new pull request` button, located on `GitHub`, or through command line or shell.

C. Results

A visualization of the network is a necessary method to ensure the layers of the network are constructed to analyze the dataset used during training. To get started we must first choose an image to pass through the network. Figure 1 shows the image chosen to pass through the network.

Figure 2 shows the image as it passes through the first hidden layer of the network. We can see the image is beginning to distort, but we can still determine what the image is.

Figure 3 shows the image as it passes through the second hidden layer of the network. The image is now distorted even more as it works it way through the network.

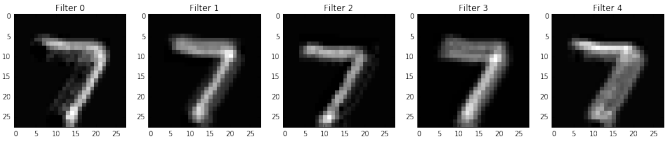

Fig. 2. First Hidden Layer

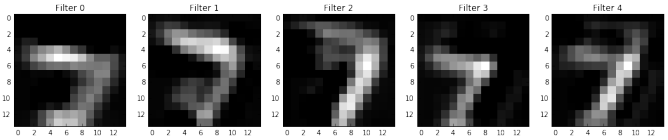

Fig. 3. Second Hidden Layer

Finally, Figure 4 shows the image as it passes through the third hidden layer of the network. At this point the network is passing the image into and out of 20 filters in order to capture all the data necessary for training. This demonstrates the "deep" context of deep learning.

As stated in the previous subsection, the results are split into four separate scenarios to measure both the number of steps to convergence and the accuracy at convergence. Table I below summarizes the number of steps, max accuracy, and step number at which max accuracy was obtained.

The first iteration of 1000 steps has a high accuracy rate at 900 steps, with a max accuracy calculation of 0.984. The next iteration was at 2500, with a max accuracy calculation of 0.988 at 2200. The third iteration consisted of 5000 steps, a max accuracy calculation of 0.9918 at 4800 steps, and finally, the last iteration consisted of 10000 steps, and obtained a max accuracy calculation of 0.9928 at 8400 steps.

Fig. 4. Third Hidden Layer

TABLE I
STEPS AND MAX ACCURACY

Steps	Max Step#	Accuracy
1000	900	0.984
2500	2200	0.9888
5000	4800	0.9918
10000	8400	0.9928

The main observation of the results is that as the model steps increased, and stepped through the data, the model tended to learn at a slower rate, but produced results with greater accuracy. That said, given a larger machine, faster processor, and continued training on the the dataset as a whole, I am confident the convolutional neural network would converge to near 100%. As seen in Table I, the number of steps increased, for each interation, in order to reach max accuracy. This makes sense, given the model has more steps to process, and essentially more time to learn.

V. CONCLUSION AND FUTURE WORK

Image processing with convolutional neural networks was a ground breaking rediscovery, applying long forgotten algorithms and theories to a new and interesting problem. The speed and accuracy of convolutional neural network makes the algorithm a prime candidate to apply in the fields of cybersecurity, threat detection, facial recognition, as well as physics and quantum computing and engineering. A practitioner with a laptop and an internet connection can calculate on large datasets, without having to consider server time and availability.

I believe we are only scratching the surface of what deep learning and convolutional neural networks can do. As more and more data becomes available, and with more research and experimentation, advancing deep learning will become easier and more valuable for everyday life.

ACKNOWLEDGMENT

I would like to thank the students in Old Dominion University's MSIM 607 machine learning class. The dialogue and discussions aided in my comprehension of the information; Melissa Rich for assisting with proofreading this paper; Dr. Li for his patience and understanding; my staff for offering ideas to improve and optimize the code, and my family for letting me write in peace and quiet.

References

Ambati, S., and C. Click. n.d. “H2O Documentation.” Available at https://www.h2o.ai (2018/04/28).

Anaconda. n.d. “Anaconda Distribution 5.” Available at https://docs.anaconda.com/ (2018/04/28).

Bishop, C M. 2006. *Pattern Recognition and Machine Learning*. Cambridge CB3 0FB, U.K.: Springer. https://www.springer.com/us/book/9780387310732.

Bottou, L. 2010. “Large-Scale Machine Learning with Stochastic Gradient Descent.” In *Proceedings of 19th International Conference Computer Statistics*, 177–86. Princeton, NJ: Springer.

Dwivedi, P. 2018. “Handwriting Recognition Using Tensorflow and Keras.” *Towards Data Science*. January. https://towardsdatascience.com/handwriting-recognition-using-tensorflow-and-keras-819b36148fe5.

Goodfellow, I, Y Bengio, and A Courville. 2016. *Deep Learning*. 1st ed. Cambridge, MA: MIT Press. http://www.deeplearningbook.org.

keras-team. n.d. “Keras Documentation.” Available at https://keras.io (2018/04/28).

LeCun, Y, L Bottou, Y Bengio, and P Haffner. 1998. “Gradient-Based Learning Applied to Document Recognition.” In *Proceedings of the IEEE*, 86:2278–2324. 11. http://yann.lecun.com/exdb/publis/pdf/cox-98.pdf.

Sutskever, I, J Martens, G Dahl, and G Hinton. 2013. “On the Importance of Initialization and Momentum in Deep Learning.” In *Proceedings of the 30th International Conference on Machine Learning*, 1139–47. ICML.

TensorFlow. n.d. “API R1.8.” Available at https://www.tensorflow.org/api_docs/python/ (2018/04/28).