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 Networks (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 example. The dataset is a subset of the larger set available from National Institute of Standards and Technology [1]. 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 converage 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 preform image processing, whether high quality, digital examples, or hand written notes, presented using a standard office scanner, the machine learning practioner would have to extract language dependent features like curvature of different letters, spacing, black and white letter, etc., only to use a classifier such as Support Vector Machine (SVM) to distingish between writers [2]. With the publication of (LeCun et al. 1998), the analysis of handwritten, variable, 2D shapes with Convolutional Neural Network was shown to outpreform all other techniques [1].

I will show that given the advance in Application Program Interface frameworks, such as TensorFlow [3], Keras [4], H2O [5] as-well-as others, have provided not only machine learning researchers and practioners the ability and tools to quickly and efficiently analyze larges amounts of data, with what are traditionally thought of as mathematically complex, but also overly expensive, both runtime and monetarily.

The key observation in this study was, 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 hardware, and an i5 Intell 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 rollup 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 thar 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 furhter the advancements, demonstrated within this paper.

II. Related Work

A. Foundational Work

LeCun et al. (1998) laid the foundation ground 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 fucntion 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 parameter. 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 ther 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 [1]. Moreover, LeCun et al. (1998) note: A poplar minimization procedure is the stochastic gradient algorithm, also call 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 fluctutates around an average trajectory, but usually converages considerably faster than a regular gradient descent and second order methods on large training set with redundant sample...[1]. 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 carmers, with great pixel quality, and pixels pre-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 [1], [7], [8], creates thousands of weights. Furthermore, using a fully-connected network negates the fact that neightboring pixels are more coorelated that non-neighboring pixels [7].

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 know grid-like topology [8]. 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. Convultional 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 simpliest form, a convolution is an operation on two functions of a real-value argument [10].

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 sometime referred to as the feature map [10], 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 a usually the case [10]. 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 the discrete data structures [10]. It is worth noting that the major difference in the continuous form and the discrete form is; for discrete data, $x(t)$ can only take on intergers values [1],[10].

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 the section I., convolutional networks provided machine learning practioners the ability to analyze large dataset, with complex data structures, on what amounts to commodity hardware. The last point I want to make regarding convolution, is the construct that assist convolution in making the aforementioned true. Convolution neural networks, by-way of the mathematical properties of convolution, employ a sparse interactions connectivity model [10]. Unlike typical neural networks that employ a full contected architecture, convolution 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 improved, and computing the output required fewer operations [1],[6],[10]. A comparison of the time complextity 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 connection to k, where $m > k$, reducing the required number of parameters to kxn, and a reduced time complexity to $O(k$x$n)$.

B. Deep Learning:

IV. Experiment

A. Dataset

The dataset used for the study in the MNIST [3], extracted using TensorFlow. The dataset used for this study, is a subset of a much larger dataset, orignally made available by NIST [1]. 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` funciotn 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 used 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), within the Spider IDE, and a user-defined virtual environment. The code does require python $>$ 3.5.

C. Results

V. Conclusion and Future Work

The conclusion goes here.

Acknowledgment

The author would like to thank Melissa Rich for assisting with proofreading this paper. Dr. Li for his paitence and understanding, my staff for offereing ideas to improve and optimize the code, and my family for letting writing in piece and quit.

References

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

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

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.