Optimizing Convolutional Neural Networks for Text Analysis

Madison Jordan
California State University, Fullerton
email: madisonjordan@csu.fullerton.edu

Abstract

Deep Learning Neural Networks simulate neurobiological features that allow learning new concepts through comparison and differentiation to form weighted connections between them through processing information and building an understanding over time, much like a human would [1]. Programming machines to automate more tasks, which can be performed faster or cheaper than the same task performed by a human. With further development, this would allow for more accurate machine translations, object or handwriting recognition, as well as the sentiment analysis of text or similar text classification task.

At the time of this writing, machine learning is not able to perfectly replicate the level of sophistication the ability of the human brain to efficiently learn at a finer granularity and perceive subtle distinctions; development evolutionary algorithms that might allow machines to more closely mimic human learning are still in their infancy stage. This paper focuses on methods machine learning models use to sufficiently emulate neurons and the forming of pathways through optimization in performing better in classification or regression tasks upon image and text datasets, which will be further modified for research upon improving accuracy in a pre-trained model for specialized text analysis tasks.

I. Introduction

A. Background

One of the main tasks of machine learning is to generalize previously acquired information, apply that understanding to new information and accurately identify or categorize the concept in a more efficient manner than could be performed by humans [2] Depending on the intended use of the task, an improvement might not be limited only increased accuracy; it be a tradeoff between accuracy, time, or cost – which may include hardware limitations.

Convolutional Neural Networks (CNN) are classification algorithms that learn from labeled datasets to be able to classify similar data as the correct label; this understanding is built through a collection of "neurons" formed into layers to learn the distinguishing features of the concept it is learning [3, 4]. As the model learns, the weights of the features learned in the feature map, or kernel, are adjusted with each data point seen to create the "bias" on which the model will use to assign labels to unseen data. [1, 3]

The advantage in convolutional neural networks for deep learning that allow a network to require less space for parameters due to its properties of parameter sharing, sparse connections, and equivariance [1]. Unlike in the use of fully connected layers which do not have sparse connections, every

input does not directly influence every output as represented by arrows in Figure 1, as illustrated in [1, Figure 9.14]. Parameter sharing allows the weight of one input to be shared among the other inputs in the network, which allows the property of equivariance, where the change in one input will have an effect on the output [1, 3].

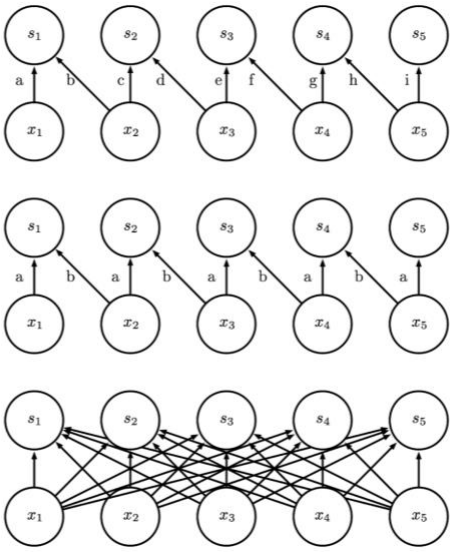

Figure 1. Comparison of local connected layers, convoluted layer, and fully connected layers, as illustrated in [1, Figure 9.14]

B. Object Recognition in Images

Convolutional Neural Networks allow the recognition of images or text even after small translations in the input, a property called invariance, using methods such as pooling, which might not otherwise be accurately identified when using other machine learning algorithms [1, 4]. This makes it useful in practical applications with datasets in which images might be at varying angles depending on the angle the camera was pointed at the time the subject was captured. [4]

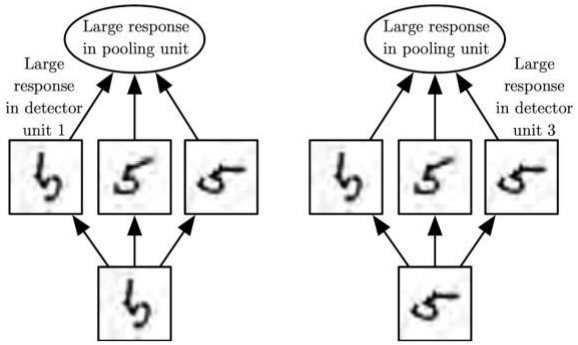

Figure 2. Example of learned invariances in handwriting analysis, as illustrated in [1, Figure 9.9]

C. Sentiment Analysis in Text

As mentioned in [4], the use of convolution with pooling works better with image recognition than it does with analyzing the meaning of sequences of words, where the location of the word in the sentence would have impact on the meaning, whereas in a picture a single pixel, being in a slightly different location might only mean it's at an angle rather than change the understanding of the image itself. In [5], the use Convolutional Neural Networks and Long Short-Term Memory (LSTM) is shown to improve the accuracy of text analysis when determining the overall sentiment of reviews on IMDB, which uses LSTM to train the model to understand the differences in meaning as a result of the sequences of the words in the reviews.

II. Approach

Due to the relatively new research on machine learning algorithms the constant development of variations to mitigate potential computational cost, such as the ones detailed in [1], current models applied to image or text analysis might be missing optimizations detailed in recent deep learning publications. These techniques may be required to develop pre-trained models that can perform better than publicly

available generalized models on more specialized tasks, as shown in optimizations of text sentiment analysis [5].

A. Convolutional Network Layer

A layer in a convolutional network goes through three stages: a convolution stage, detector stage, and a pooling stage.

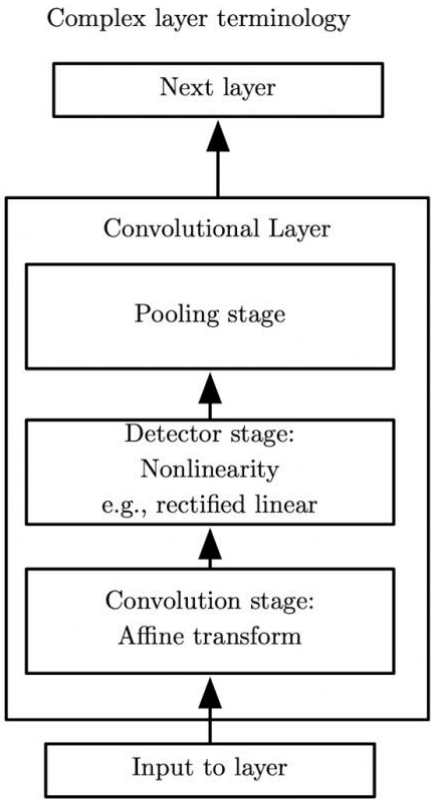

Figure 3. example implementation of Convolutional Neural Network layers, as illustrated in [1, Figure 9.7]

During the convolution stage, the layer produces a set of linear activations through convolutions; these activations are then each passed to a nonlinear activation function. [1] Finally, the pooling stage makes the model invariant to small translations of the input, allowing for more flexibility of whether a feature is present rather than whether its location precisely matches that of the learned location from the training data. [1]

$$s(t) = \int x(a)w(t-a)da.$$

$$s(t) = (x * w)(t).$$

Figure 4. Basic Convolution Function. x is the input, while w is the output, or kernel, as shown in [1, Figure 9.1 and 9.2]

B. LSTM

As a machine learns, it may forget some of its gained understanding of a concept as it takes in more information if older inputs are not reintroduced, known as catastrophic forgetting. [1] Recurrent neural networks (RNNs) such as Long Short-Term Memory (LSTMs) allow for the knowledge of previous inputs to be kept which may have an effect on the output [1, 5].

Due to the importance of the sequence or order of words for understanding meaning, LSTMs would allow for the model to more accurately interpret the meaning of text from the context, or meaning of preceding text, rather than the limited context of the word it is on.

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma \left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)} \right)$$

$$f_i^{(t)} = \sigma \left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)} \right),$$

$$g_i^{(t)} = \sigma \left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)} \right)$$

Figure 5. LSTM state for time t and cell i, where b is the bias, U is the input weights, W is the recurrent weights. $x^{(t)}$ is the current input vector, $h^{(t)}$ is the current hidden layer vector, which contains all the outputs of all of the LSTM cells. f is the forget gate formula and g is the output gate formula, as shown in [1, Figure 10.41, 10.40, 10.42]

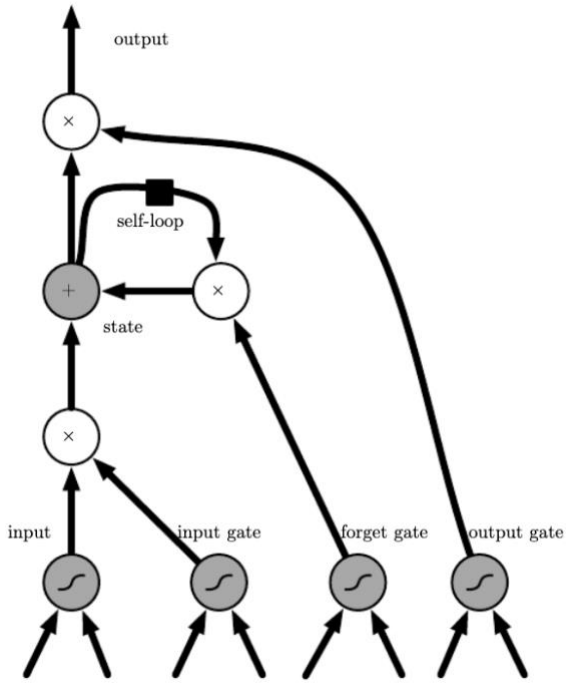

Figure 6. diagram of LSTM using forget gates to regulate weights of previous inputs that are reintroduced into the model, as illustrated in [1, Figure 10.16]

C. Optimization

Optimization in deep learning is intended to be used to indirectly optimize a performance measure indirectly through the reduction of a cost function. [1]

Stochastic Gradient Descent (SGD) is the most used optimization method in deep learning. [1] SGD allows for the possibility model to distinguish between concepts with a large degree of accuracy without processing all of the training data in large datasets, "The most important property of SGD and related minibatch or online gradient-based optimization is that computation time per update does not grow with the number of training examples." [1].

Algorithm 8.1 Stochastic gradient descent (SGD) update

Require: Learning rate schedule $\epsilon_1, \epsilon_2, \ldots$
Require: Initial parameter $\boldsymbol{\theta}$
  $k \leftarrow 1$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \hat{\boldsymbol{g}}$
    $k \leftarrow k + 1$
  end while

Figure 7. Algorithm for SGD, as shown in [1, Section 8.3]

D. Regularization

Regularization (parameter norm penalty) allows parameters' weight vectors to pull closer to their true values by using penalty functions to shrink the weight vectors, known as weight decay, as the machine trains. [1]

L_2 regularization, also known as ridge regression or Tikhonov regression, is the most common form of regularization used. [1] It allows the machine to penalize the weights of features which seem to have a lower impact on the output by allowing the machine to perceive the input of the model to have a higher variance and penalize any inputs whose covariance with the output target is low in comparison. [1]

L_1 regularization assumes that some parameter weights have an optimal value of 0, meaning they can be discarded as they have no impact on the output, thereby simplifying learning so that only the features with the most impact can be selected. [1]

L_2 Regularization $\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$

L_1 Regularization $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 = \sum_i |w_i|,$

Figure 8. Regularization for in L_2 and L_1, as shown in [1, Section 7.1]

III. Implementation

Using a pre-labeled sarcasm dataset, I hope to implement a model using convolutional neural network combined with an LSTM, as used in another text analysis dataset for analyzing sentiment of reviews in IMDB [5]. By starting from the model used in the previous text analysis study with an ReLU activation function, L_2 regulizer, and a learning rate of 0.01 with decay, it might be possible to produce reasonable results; however, the impact of using the optimizer RMSprop over Adam or SGD was minimal. [5] These hyperparameter values, such as the optimizer, would need to be adjusted to find the optimal model to fit the dataset [1,5].

A. Example Implementation

Using Keras and Tensorflow on DirectML with an AMD GPU, I followed a guide to create an interactive webapp to compare the different effects changes in hyperparameters would have on the performance of a model that used the Cifar10 image dataset. [6] The visualization of the interactive web app using the streamlit library used in the example would help in determining which adjustments to make to optimize the model.

IV. Conclusion

A. Current State

Progress on the sarcasm dataset itself has not been made, but the problems I had over the summer in getting Tensorflow to run at all have made significant progress in allowing me to move forward in implementation, albeit in less than ideal conditions for testing.

B. Next Steps

The first models for the sarcasm dataset need to be implemented and analyzed. A recent method for training models such as Google's BERT might help reach optimal performance.

I will need to practice using the relevant machine learning libraries for this type of dataset and become familiar with the different dependencies' compatibility.

C. Challenges

After failing to get the Tensorflow machine learning library running over the summer, I discovered Tensorflow officially only supports NVIDIA graphics cards using CUDA. Due to lack of access to a computer with a NVIDIA graphics card or to the research lab at school, I had to find workarounds to get the Tensorflow library to work on any of the computers I have available.

At the time of this writing, the AMD ROCm implementation of Tensorflow is in early development and has hardware limitations and runs only on Linux. There were difficulties in the running another Tensorflow alternative PlaidML.
I found success in the recent release of DirectML by Microsoft, which is still in early development and does not support Tensorflow 2 at this time.

Running code from examples which use deprecated dependencies or those with new package names has also been a challenge in importing the necessary python libraries.

V. Acknowledgements

I'd like to thank my mentors Dr. Doina Bein of California State University, Fullerton and Dr. Abhishek Verma of New Jersey City University for guiding me through this

material and creating realistic expectations for myself in this program. I'd also like to thank Dr. Brianna Blaser and Dr. Richard Ladner of UW, as well as my family and friends for encouraging me to apply to the DREU this summer despite my hesitance. Thank you to all the DREU organizers and facilitators who were understanding during this difficult time and openly accommodating to support in any way they in which they were able.

References

[1] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, Available: http://www.deeplearningbook.org.

[2] P. Domingos, "A few useful things to know about machine learning", *Communications of the ACM*, vol. 55, no. 10, pp. 78-87, 2012. Available: 10.1145/2347736.2347755 [Accessed 6 August 2020].

[3] J. Wang, R. Turko, O. Shaikh, H. Park, N. Das, F. Hohman, M. Kahng, and P. Chau, "CNN Explainer," *Polo Club of Data Science @ Georgia Tech: Human-Centered AI, Deep Learning Interpretation & Visualization, Cybersecurity, Large Graph Visualization and Mining*. [Online]. Available: https://poloclub.github.io/cnn-explainer/

[4] A. Bhandare, M. Bhide, P. Gokhale and R. Chandavarkar, "Applications of Convolutional Neural Networks", *Api.semanticscholar.org*, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:45888197

[5] A. Yenter and A. Verma, "Deep CNN-LSTM with Combined Kernels from Multiple Branches for IMDb Review Sentiment" presented at the 8th **IEEE** Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), Oct. 19-21, 2017, New York, NY, USA. Available: https://vermaabhi23.github.io/publication/2017UEMCON1.pdf

[6] V. Alto, "Interactive Convolutional Neural Network," *Medium*, 28-Oct-2019. [Online]. Available: https://medium.com/dataseries/interactive-convolutional-neural-network-65bc19d8d698