Image Recognition: Adam's Method and Convolutional Neural Networks

Sichen Zhong

UC Santa Barbara
sichenzhong@ucsb.edu

Abstract

This paper analyzes the usage of Adam's Method and Convolutional Neural Networks (CNN) in optimizing visual classifications. Adam's Method is a stochastic gradient-based optimization algorithm that utilizes adaptive estimates of stochastic functions and replacement optimization in order to train deep learning models. This method is well-suited for big data with many parameters since efficient stochastic optimization techniques are required to account for noisy objectives. Furthermore, CNN architecture is composed of many unique layers conducive to detecting different features in an image. The accuracy of these numerical methods will be analyzed through sampling and initiating a loss function.

Part I. **Problem and Model Formulation**

1. *Convolutional Neural Network (CNN)*

In our study, utilizing CNN is more preferential than Regular Neural Networks since CNN explicitly assumes all inputs are images and subsequently, conforms the architecture to be comprised of "neurons" arranged in 3 dimensions: width, height, and depth. Our study focalizes on the FashionMNIST image data set, which has dimensions 28x28x1 due to its fixed resolution of 28-by-28 pixels and its grey-scaled attribute that employs a one-color RGB channel.

A CNN is comprised of a multi-layer architecture, first commencing with an input layer and ultimately producing an output layer. Since CNN is also a feed-forward neural network, the series of layers composed between the input and output layers are hidden by the activation function and the final convolution. Albeit hidden, they are pivotal to the performance of the CNN in part to its embodiment of "neurons" that receive inputs and learn weights/biases. One such layer will be the convolutional layer, which is commonly initiated by a Rectified Linear Unit (ReLU) activation layer and is subsequently coupled with a pooling layer. Optionally, a normalization layer can be implemented to normalize the activation layer to a standard scale through batch normalization or layer normalization. The final convolution involves backpropagation, which weighs the accuracy of the end product by computing errors in reverse order. Hence, the architecture of CNN is modeled as:

(1) **Input**, *input layer* retrieves images from the imported image data set as 28x28x1 values;
(2) **Convolutional Layer**, *building block of CNN* computes the dot product between the neuron weights and the selected regions they are connected to within the input layer;
(3) **ReLu Layer**, *activation function* elementwise non-linearity applied to conv. layer;
(4) **Pooling Layer**, *spatial resizing* reduce parameters/computations to control overfitting;
(5) **Fully-Connected Layer**, *output layer* softmax function to compute class scores (prob.);

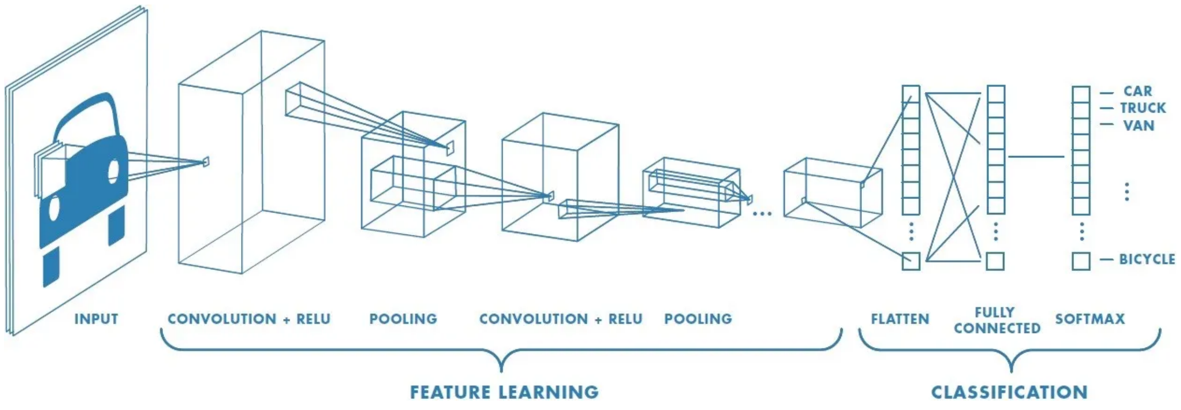

FIGURE 1. *CNN Architecture*

PART II. **Theory Supporting Numerical Methods and Validations**

2. *Adam's Method*

2.1. *Theory*

Adam's Method draws upon stochastic gradient descent in order to compute adaptive "neuron" weights that facilitate the computation of non-convex problems efficiently. These "neuron" weights are construed and updated through each iteration of "adaptive moment estimation" during the data training process. Its efficiency can be attributed to the advantageous qualities of two other extensions of stochastic gradient descent that it borrows ideology from:

(1) **Adaptive Gradient Algorithm**, *AdaGrad* maintains learning rates unique to each parameter by compiling past observational features, parameters are updated based on relevancy and not frequency, improves performance for computations with sparse gradients (e.g. natural language processing and computer vision);
(2) **Root Mean Square Propagation**, *RMSProp* adapts learning rates unique to each parameter by computing the decaying average of recent magnitudes of partial gradients, specifically dedicated to non-stationary computations (e.g. noisy);

Adam's Method incorporates the above qualities by maintaining an unique learning rate for each parameter that is consistently adapted based on the estimates of the mean (the first moment of the gradient) and the un-centered variance (the second moment of the gradient). This is attained by calculating the Exponential Moving Average (EMA) of the gradient and the squared gradient on the current mini-batch gradient descent, which places more weight on recent data and thus, allows the stochastic gradient descent to follow the trend more tightly and enables the "neurons" to react quicker. The parameters β_1 and β_2 are hyper-parameters that serve as stabilizers, with an initialization value of close to one which results in an initial bias of around zero. This is conducive to easier replacement of the initial bias with the biased estimates of the moving averages and subsequently, the bias-corrected estimates of the moving averages before updating the learning parameters. As a result, the decaying rates of the moving averages are stabilized and the hyper-parameters are given "interpretive intuition" that requires little manual/computational tuning from the coder.

```
1: procedure ADAM'S METHOD ALGORITHM
2:     Parameters ← $\alpha$(Stepsize); $\beta_1, \beta_2 \in [0,1)$(Exponential decay rates); $f(\theta)$(Stochastic function); $\theta_0$(Initialization)
3:     $m_0 \leftarrow 0$ (Initialize 1st moment vector)
4:     $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
5:     $t \leftarrow 0$ (Initialize time step)
6:     while $\theta_t$ does not converge: do
7:         $t \leftarrow t + 1$
8:         $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
9:         $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
10:        $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
11:        $\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
12:        $\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
13:        $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
       return $\theta_t$ (Resulting parameters)
```

FIGURE 2. *Adam's Method Pseudo-Code*

2.2. *Validating Adam's Method as an Optimization Algorithm*

A well-known optimization algorithm also utilized in deep learning is the Classical Stochastic Gradient Descent (SGD). This optimization algorithm utilizes steepest descent to compute a single learning rate termed alpha, that is applied to all "neuron" weights as an update and remains unchanged throughout the training process. Some other sub-versions of SGD are:

(1) **Batch Descent**, *utilizes entire data set* accuracy in direction but heavy in computation;
(2) **Single-Batch Descent**, *utilizes single example* randomly chosen examples for each iteration, faster computation but inaccuracy in direction;
(3) **Mini-Batch Descent**, *utilizes small sub-set* randomly chosen subsets per iteration;

While SGD may prove advantageous in producing better generalizations, it is not an optimal algorithm since its application of only one single global learning rate for all parameters creates extraneous noise that consequently procures inaccurate approximations of the gradient. In other words, SGD is locally unstable since it tends to ignore better approximations and gravitate towards minimas and saddle points. Furthermore, SGD is computationally heavy and inefficient, which leads to massive memory usage. Its flaws can be summarized as:

(1) **Gravitation Towards Minimas** inhibits from finding a better solution;
(2) **Saddle Points** gradients valued at zero but respective point is not a minima;
(3) **Global Learning Rate** high learning rate correlates to high fluctuations and low learning rate correlates to slow convergence, requires manual tuning and adjustments;
(4) **Slow Progression** badly scaled data set leads to slower progression nearing solution;

In comparison, Adam's Method is a more proficient alternative because the parameter learning rates are adapted in real-time based on the estimates of the first and second moments of the gradient. Since this computation evaluates the exponentially decaying average of the last w gradients (whereas SGD only utilizes the current gradient), it stimulates more momentum within the algorithm than SGD. Furthermore, Adam's Method requires little computational memory usage and can be easily applied to large data sets with noisy/sparse gradients since the aforementioned EMA computation forces gradients to be "tighter" with each learning iteration. Its advantages can be summarized as:

(1) **Unique Learning Rates**, *faster learning* parameters that would usually be overlooked can receive the needed updates, automatic adjustments means less manual tuning;
(2) **Momentum**, *faster convergence* a certain fraction of the most current update is applied to find the next update which contributes to a faster convergence rate and ensures updates move in a particular direction, dampens oscillations and fluctuations which increases progression to the solution, gradients "tighten" with each iteration;
(3) **Less Memory Usage**, *faster computations* straightforward and efficient;

Part III. **Numerical Results from Experimentation**

3. *FashionMNIST Data Set*

3.1. *Experimentation Process*

The experimentation is initialized by loading the training set of FashionMNIST as tensors in PyTorch. Subsequently, a 2-layer CNN is implemented to detect image features. The first layer is composed of a 16 channel 2D-convolutional layer with a 5 pixel size kernel, a ReLU non-linearity activation function, a batch normalization layer, and a max pooling layer with a 2 unit size kernel. The second layer is composed of a 32 channel 2D-convolutional layer with a 5 pixel size kernel, a ReLU non-linearity activation function, a batch normalization layer, and a max pooling layer with a 2 unit size kernel. This produces $7 * 7 * 32 = 1568$ features that is feeded into a fully-connected layer, which contains a softmax activation function that assigns probabilities to the respective classification categories. Finally, the model is optimized by Adam's Method and trained by minimizing the cross-entropy loss function.

Consequently, the 2-layer CNN produces the following loss history plot. While there are fluctuations in the slope of the loss plot, the general downward trend in loss for each increasing epoch indicates comparable performance.

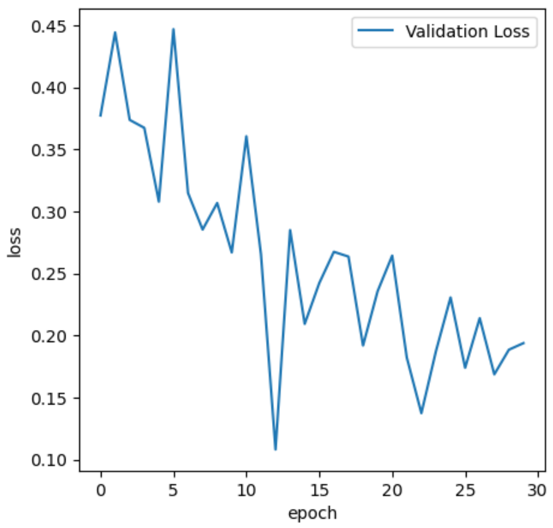

Figure 3. *Loss History Plot*

3.2. *Results and Accuracy*

To further facilitate our experimentation, a testing set of FashionMNIST, separate from the training set, is loaded into the CNN. This closely follows the principles behind supervised learning, which proposes that models should be trained on separate initial data first before being fed new data that it needs to analyze. As a result, the model is enhanced since the information retrieved from the training data is utilized as foundational knowledge to test new data. Consequently, our CNN produces a resulting accuracy score of around 90% in classifying the 10000 test images. Hence, it is sufficient to conclude that the model is relatively accurate.

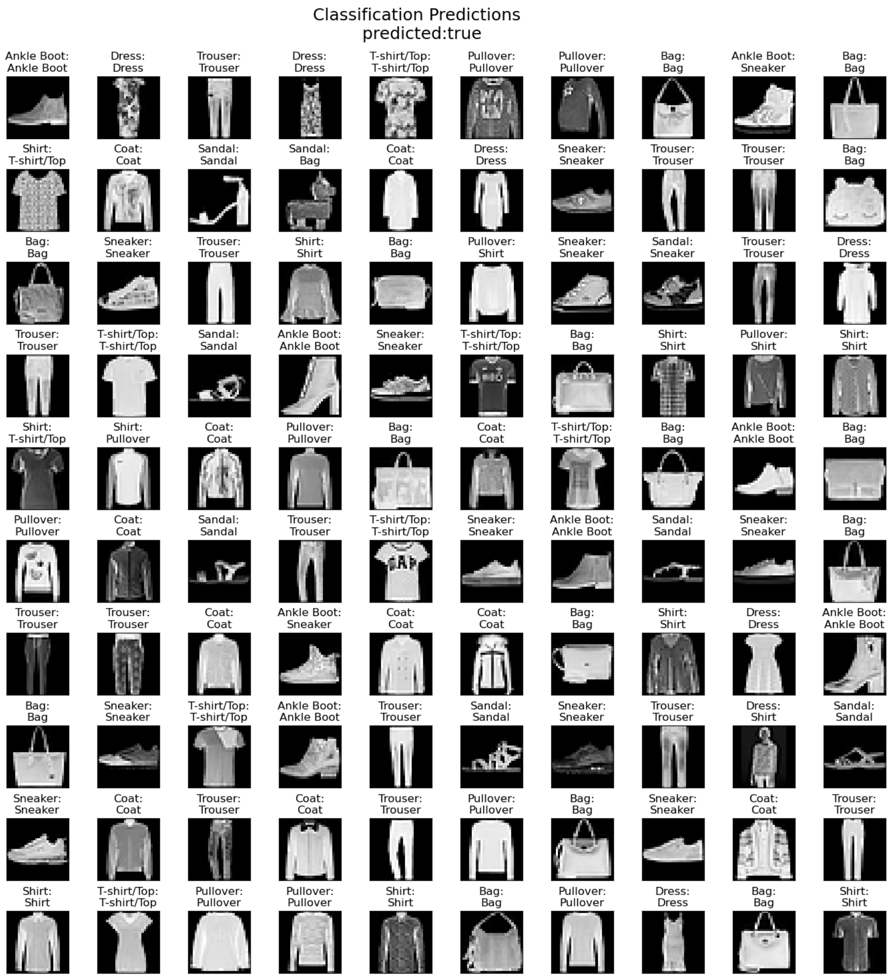

FIGURE 4. *Results Table*

References

1. BA, J. L., KIROS, J. R., and HINTON, G. E., 'Layer Normalization', *arXiv preprint* (2016) arXiv:1607.06450.
2. KETKAR, N., 'Stochastic Gradient Descent', *Deep Learning with Python* (2017) Apress, Berkeley, CA, doi: $10.1007/978 - 1 - 4842 - 2766 - 4_8$.
3. KINGMA, D. P., and BA, J. L. , 'Adam: A Method for Stochastic Optimization', *International Conference on Learning Representations* (2015).
4. LECUN, Y., BENGIO, Y., and HINTON, G., 'Deep learning', *Nature* (2015) 521(7553), 436-444.
5. Z. ZHANG, 'Improved Adam Optimizer for Deep Neural Networks', *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)* (2018) pp. 1-2, doi: 10.1109/$IWQoS$.2018.8624183.

Sichen Zhong
UC Santa Barbara, CA

sichenzhong@ucsb.edu