Artificial Neural Network for CIFAR-10 Image Classification: Effects of Network Architecture, Regularization and Dropout

Akwasi Darkwah Akwaboah*

***Abstract*— Machine learning algorithms have recently done the 'magic' at image classification. Notable algorithms include deep learning approaches where neural network implementations have been used. A major requirement for obtaining high test classification accuracies is training the network on large datasets and along with other network performance augmentation strategies like regularization and dropout. As part of my NSU coursework EEN614 (Neural Networks); this report presents a neural network implementation using Keras with CIFAR-10 datasets used for training, validation and testing. Effects of regularization and dropout on test accuracy and reducing overfitting are investigated. Code for this implementation can be accessed via this github link**

***Index Terms*— machine learning, neural networks, image classification, CIFAR-10 dataset**

I. Introduction

Machine learning algorithms particularly neural networks have proved to be useful at image classification tasks. However, crucial to arriving at high accuracies is the requirement of large datasets. A popular "helloworld" dataset for implementing image classification with machine learning is the CIFAR-10 datasets[1], comprising of 60,000 32×32 pixeled natural images with 10 classes. Artificial neural network algorithm was inspired by neuroscience, with a single computational node termed perceptron as per Frank Rosenblatt[2]; typical network architecture consists of an input layer, which receives the inputs, i.e. image pixel values and an output layers selects a single node tied to an image class that the network predicts. Sandwiched between these two nodes are hidden layers containing activation functions that introduced non-linearities. Though simple neural networks have achieved decent performances, there exists several network augmentation strategies such as regularization and dropout which help the network perform better on test data. Regularization is a technique that strives to reduce the overfitting, a phenomenon that occurs when the network overlearns and generalizes to the training data while performing poorly on prediction on test data. Regularization decays the weights to allow generalization to future data. Drop out, on the other hand, employs a randomized evolutionary strategy at an optimized network connection. Here, selected connections are randomly dropped, and performance metrics accessed. In this report, python-based Tensorflow module, Keras[3] is used to implement neural network architectures in google colab environment aimed at investigating the effects of number of hidden layers, L2-norm regularization and dropout on the training, validation and testing accuracy for the CIFAR-10 datasets with reference from[4]

Akwasi Darkwah Akwaboah is with the Electronics Engineering Department, Norfolk State University

II. Data Preprocessing

Data (pixel values) was normalized from 0-255 to 0.0-1.0 and from the 3 channel RGB to gray scale using the luma coding weighted average in video systems. This was done to reduce the 3 channeled images to a single channel that can be introduced (flattened) into the network input. 10 of the CIFAR-10 datasets in RGB and their corresponding grayscale versions are shown in figure 1 below.

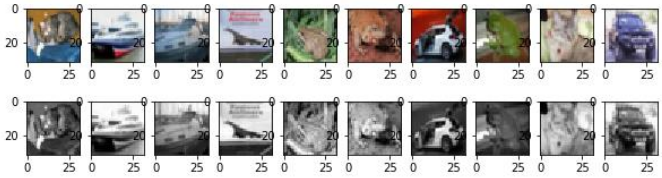

Figure 1. RGB to grayscale conversion to reduce 3 channeled images to single channel

III. Larger Network (8 hidden Layers) Performance

An initial attempt was a network architecture with 8 hidden fully connected layers each with 512 nodes and interspaced with rectified linear unit (reLU) activation functions. An output layers has a softmax activation. The network was trained with 50,000 of the datasets, 10,000 (20%) of which was used in validation. The remaining 10,000 images were used for testing. Training was done over 20 epochs with batch size of 128 using the 'Adam' optimizer[5]. Figures 2 and 3 show the training and validation accuracies and losses. Maximum training and validation accuracies at the end of the the 20th epoch are 50.91% and 41.59% respectively. Testing accuracy of 41.50% was achieved here. This significant difference (~10%) in accuracies depicts *overfitting* as validation accuracy and loss does not improve significantly after about epoch 10 even though, training accuracy and loss seem to improve and could get even better over more epochs. In summary, the network generalizes

to the training data but performs poorly on testing. These metrics form the yardstick for all other preceding network modification. The goal for implementing the next strategies is to overcome overfitting. Techniques overcoming overfitting and improve performance with lesser computation include modifying the network architecture; e.g. number of layers and their nodes, regularization and dropout.

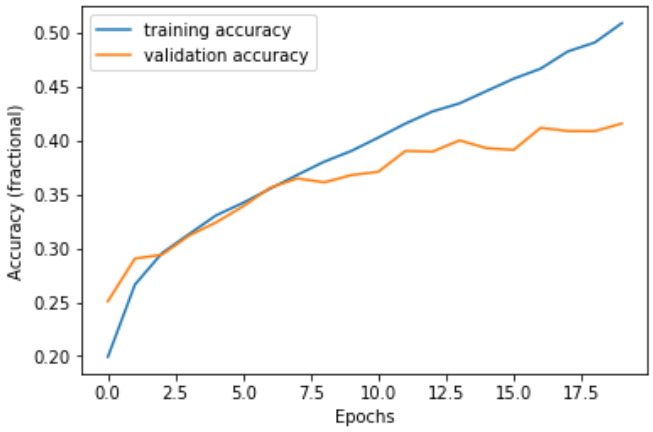

Figure 2. Training and Validation accuracies – 8-layer network

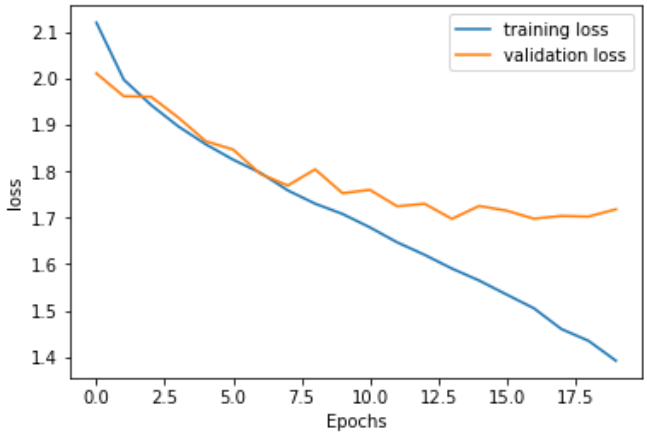

Figure 3. Training and validation loss - 8-layer network

IV. Smaller network (4 layers) – reduced nodes

Big does not necessarily mean better in neural networks as varying the number of nodes per layer in a networks with fewer layers can achieve comparable or even better performance metrics than ginormous networks. This phenomenon embodies the tenets of the "*No Free Lunch" theorem*[6]; which is paraphrased as - there is no optimal algorithm known, we can also know which algorithm(s) performs well on a dataset if we try it out. This section presents comparable performance as the previous 8-layer network with a 4-layer tapering node number architecture. Here, instead of using constant layer size as the 8-layer case; a 512-256-64-32-layer node architecture is used. Interestingly, similar accuracies are achieved with yet overfitting present. Training and validation accuracies at the end of the 20th epoch were 48.77% and 42.29% respectively. Testing accuracy was 41.51%. The difference between the training and validation accuracies was ~6%, this is however still significant overfitting and must be reduced further by techniques like regularization and dropout, which are subsequently attempted in this report.

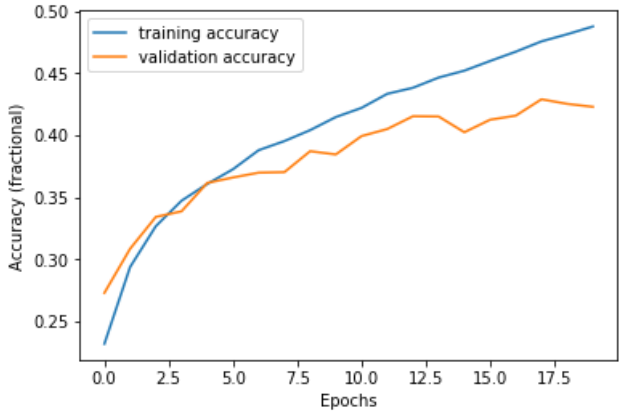

Figure 4. Training and validation accuracies - 4-layer tapering (512-256-64-32) layer nodes

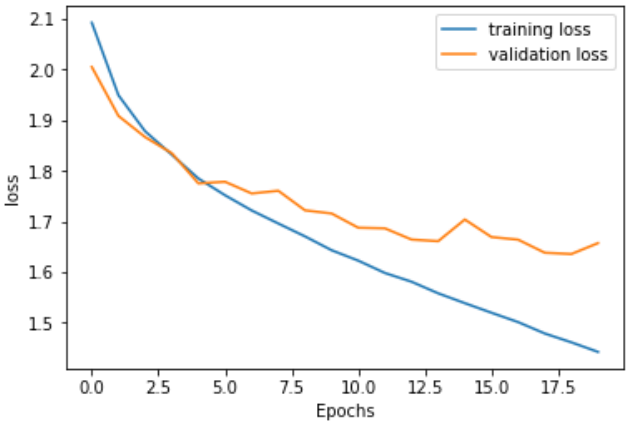

Figure 5. Training and Validation losses - 4-layer tapering (512-256-64-32) layer nodes

V. L2-Regularization

As earlier mentioned, regularization seeks to reduce overfitting by decaying the weights. In L2-norm regularization, an additional term is introduced to the loss objective function, $J(\theta|x,y)$ that introduces some inertia to its change. Mathematical formulation for this is shown in equation 1 below;

$$J(\theta|x,y) = J(\theta|x,y) + \alpha\Omega(\theta) \text{ --------- } 1$$

Where x and y are the inputs and labels for the given examples respectively, θ represents the weights, while, α is the hyperparameter for tuning the effect of the penalty term, $\Omega(\theta)$, which in this case is the L2-norm of the weights. The effects of this is seen in the training and validation accuracies and losses curves – the training and validation metrics are kept close with minimal deviation. However, the test accuracy was slightly reduced (~1% less) – 39.83%. A regularization learning rate of 0.005 was used. Training and validation accuracies achieved at the end of the 20th epoch were 42.43% and 40.21%. This result relative to the earlier implementations reduced overfitting, however, there still exist 2.6% deviation of the test accuracy from the training accuracy.

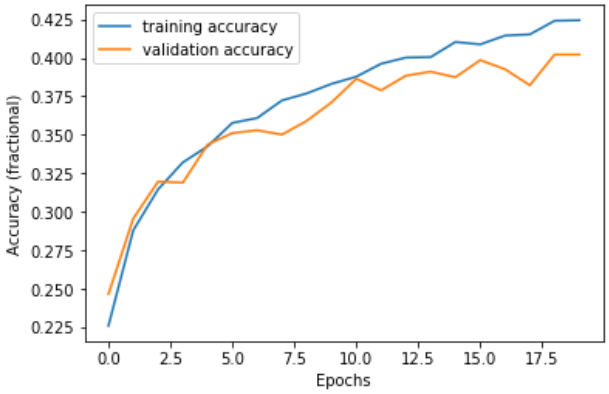

Figure 6. Training and Validation accuracies – L2 regularization

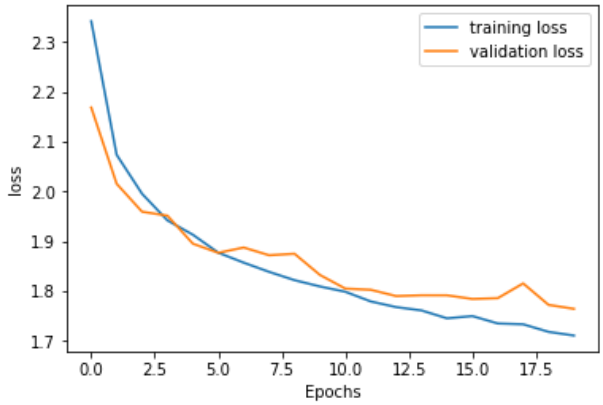

Figure 7. Training and Validation losses – L2 regularization

VI. Dropout

The idea behind dropout is to randomly eliminate a certain percentage of the network connections and evolve to find an optimized network connection. Here, a 20% dropout strategy was adopted. This was applied at the end of the hidden layers. This technique further reduces overfitting and is evident in the training and validation accuracies and losses shown in figure 8 and 9. Training and validation accuracies at the end of the 20th epoch were 40.47% and 38.79% respectively. Test accuracy achieved was 38.18%. Even though test and training accuracies did not improve. A rather reduced deviation of test accuracy from training accuracy - ~1.6%, thus a reduced overfitting was achieved with dropout

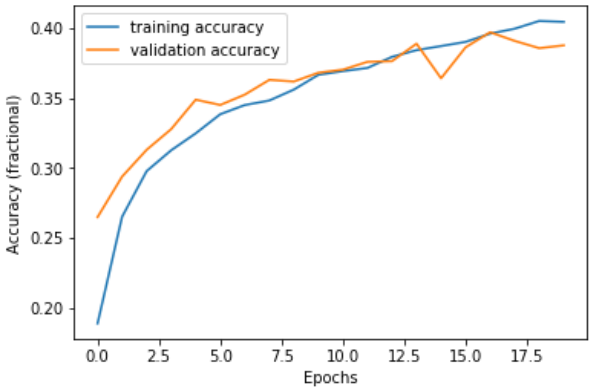

Figure 8. Training and validation accuracies - 4-layer tapering (512-256-64-32) layer nodes - drop-out

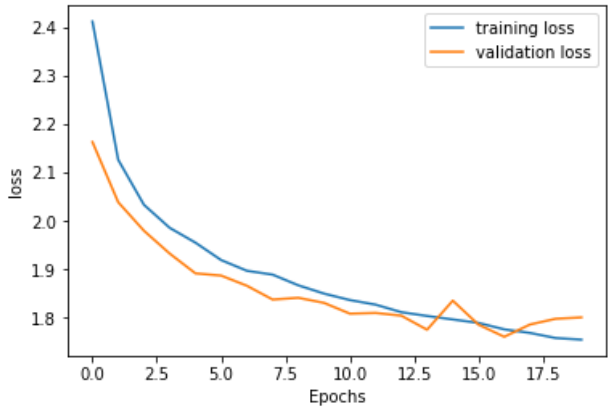

Figure 9. Figure 8. Training and validation loss - 4-layer tapering (512-256-64-32) layer nodes - drop-out

VII. Conclusion

Image classification by machine learning algorithm have recently gain popularity. Neural networks are one of many algorithms that can be used for this purpose. In this report, the effects of network architecture, L2-regularization and dropout on the performance of the network was investigated. It was noticed that large networks do not necessarily give optimum performance metrics and are most often embroiled with overfitting. L2-regularization along with dropout reduced overfitting. Code for this implementation for all four implementations are accessed via github link.

VIII. References

[1] "CIFAR-10 and CIFAR-100 datasets." [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed: 20-Oct-2019].

[2] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, Nov. 1958.

[3] F. Chollet and others, "Keras." 2015.

[4] "11_NN_Regularization." [Online]. Available: http://www.cs.nthu.edu.tw/~shwu/courses/ml/labs/11_NN_Regularization/11_NN_Regularization.html. [Accessed: 21-Oct-2019].

[5] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014.

[6] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization," 1997.