**Linearized Loss-based optimal Learning-rate**

# **SUMMARY**

Stochastic gradient methods are the driving force behind the recent boom of deep learning. As a result, the demand for practical efficiency as well as for theoretical understanding has never been stronger. Naturally, this has inspired a lot of research and has given rise to new and currently very popular optimization methods such as Adam, AdaGrad, or RMSProp, which serve as competitive alternatives to classical stochastic gradient descent (SGD)

The optimization issue has been haunting the industry since decades, especially the slow convergence rates due to the decision of step sizes being very small and thus affecting the learning rates of the algorithm. People have suggested multiple approaches to tackle this problem like having Bayesian approach instead of frequentist approach and finding the best probability for the next step size, other study suggests training the learning rates itself by using gradient decent and in practice the decaying learning rates have been very widely used and applauded.

We try to minimize the manual tuning required for the deep learning models and we provide a step size adaptation which is practical and easy to replicate and implement. Our adaptation method is called Linearized Loss-based optimal Learning-rate (L4) and it has two key features. First, it operates directly with the (currently observed) value of the loss. This eventually allows for almost independent step size computation of consecutive updates and consequently enables very rapid learning rate changes. Second, we split the two roles a gradient vector typically has. It provides both a local linear approximation as well as an actual vector of the update step. We allow using a different gradient method for each of the two tasks. At every step, this method computes the linearization of the function at the current point and proceeds to the root of this linearization. We use analogous updates to locate the root (minimum) of the loss function.

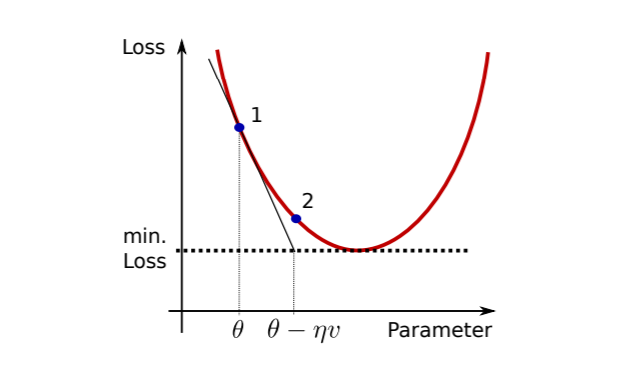
# **IMPLEMENTATION**

We tried to implement and compare the default settings of the step sizes of optimization algorithms like Adam and Momentum on the CIFAR-10 dataset and combining it with the previous experiments of CNN architecture as mentioned below:

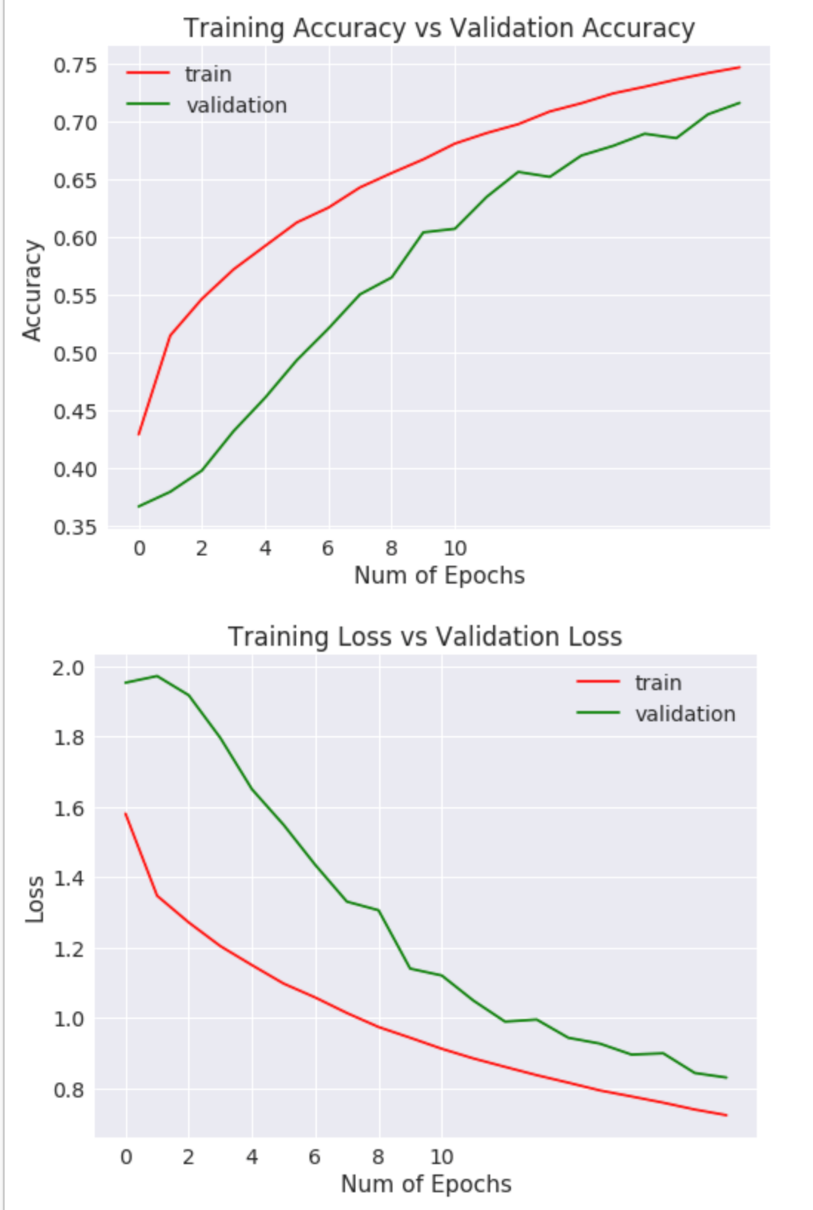
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Layer (type) Output Shape Param # ================================================================= conv2d\_31 (Conv2D) (None, 32, 32, 32) 896 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dropout\_26 (Dropout) (None, 32, 32, 32) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ conv2d\_32 (Conv2D) (None, 32, 32, 32) 9248 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ max\_pooling2d\_4 (MaxPooling2 (None, 32, 16, 16) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ conv2d\_33 (Conv2D) (None, 64, 16, 16) 18496 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dropout\_27 (Dropout) (None, 64, 16, 16) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ conv2d\_34 (Conv2D) (None, 64, 16, 16) 36928 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ max\_pooling2d\_5 (MaxPooling2 (None, 64, 8, 8) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ conv2d\_35 (Conv2D) (None, 128, 8, 8) 73856 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dropout\_28 (Dropout) (None, 128, 8, 8) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ conv2d\_36 (Conv2D) (None, 128, 8, 8) 147584 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ max\_pooling2d\_6 (MaxPooling2 (None, 128, 4, 4) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ flatten\_6 (Flatten) (None, 2048) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dropout\_29 (Dropout) (None, 2048) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dense\_11 (Dense) (None, 1024) 2098176 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dropout\_30 (Dropout) (None, 1024) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_12 (Dense) (None, 10) 10250

====================================================



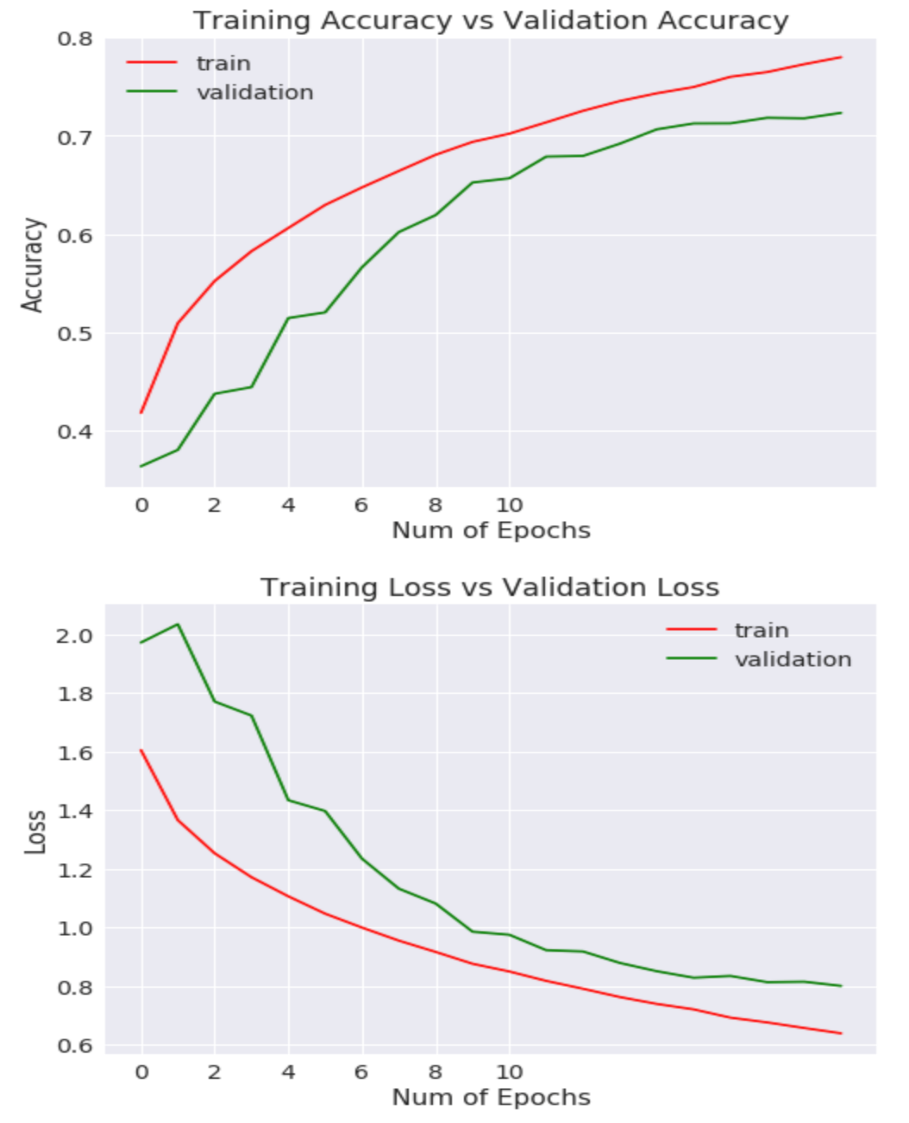
## **L4ADAM**



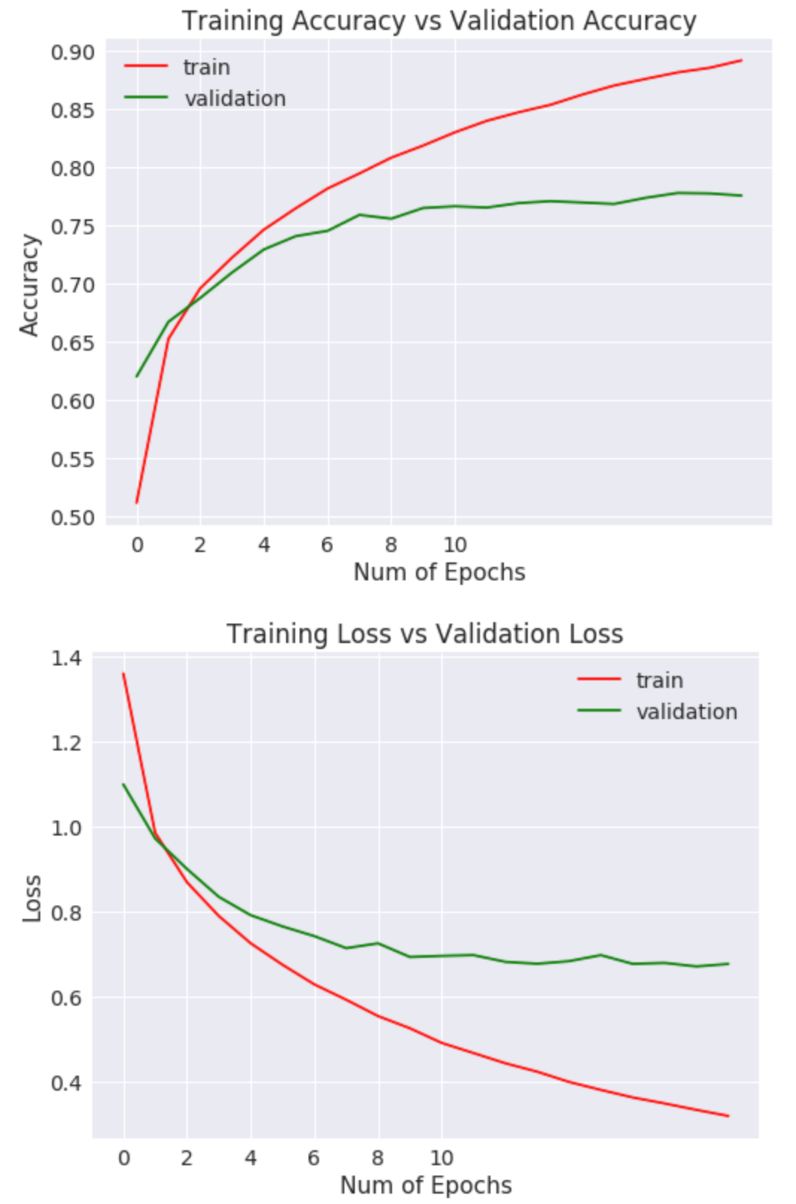
### **L4ADAM (FRACTION = 0.1)**



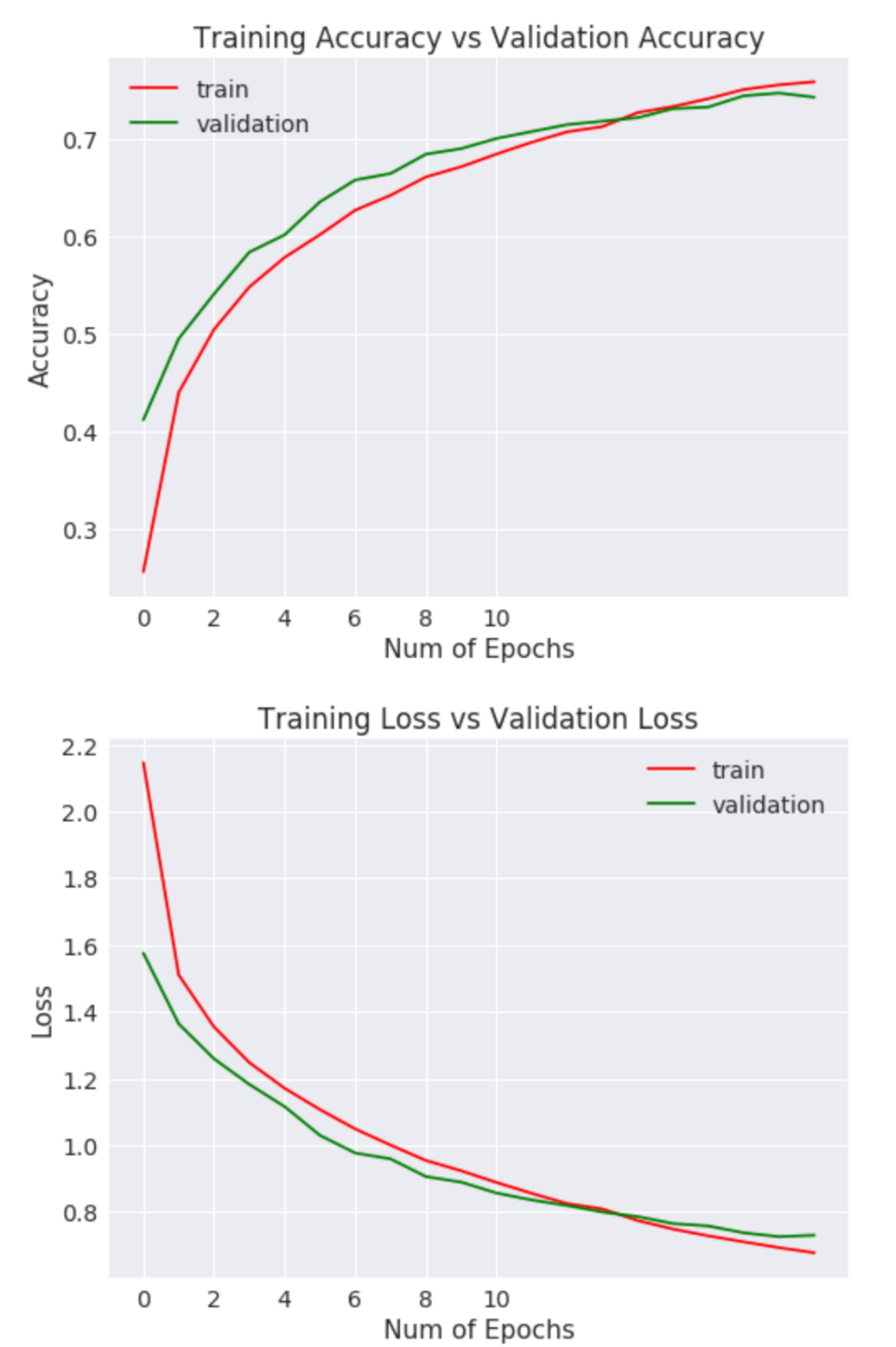
### **L4ADAM (FRACTION = 0.2)**



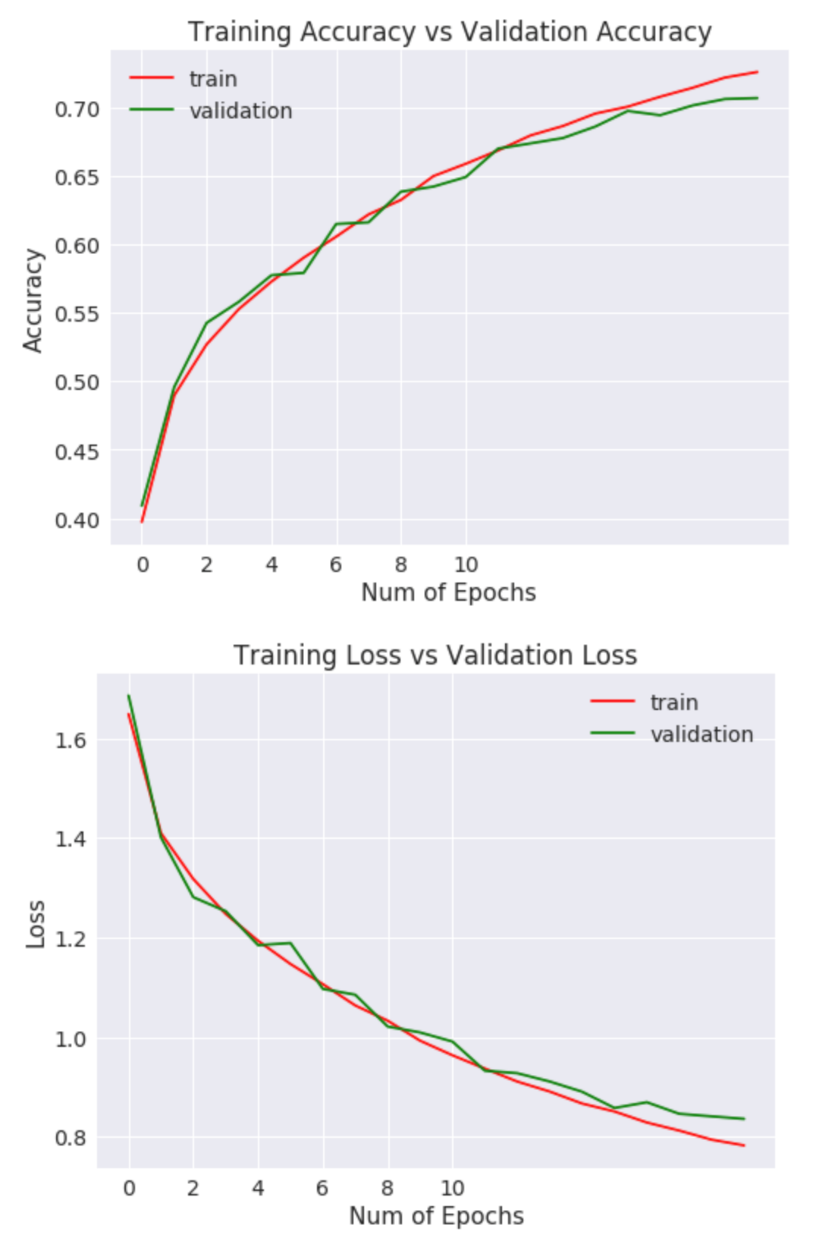
### **L4ADAM (FRACTION = 0.4)**



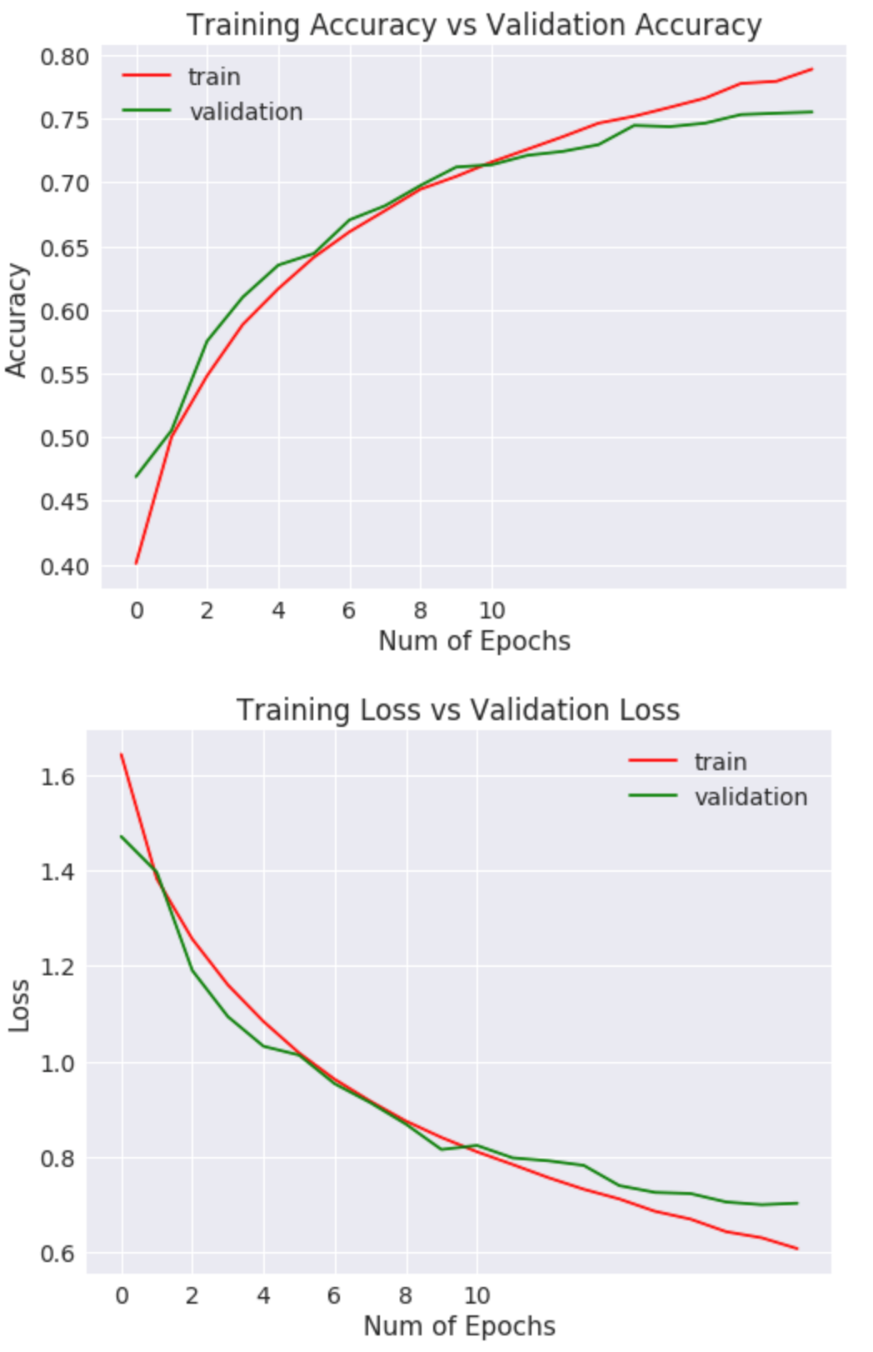
### **L4ADAM (FRACTION)**



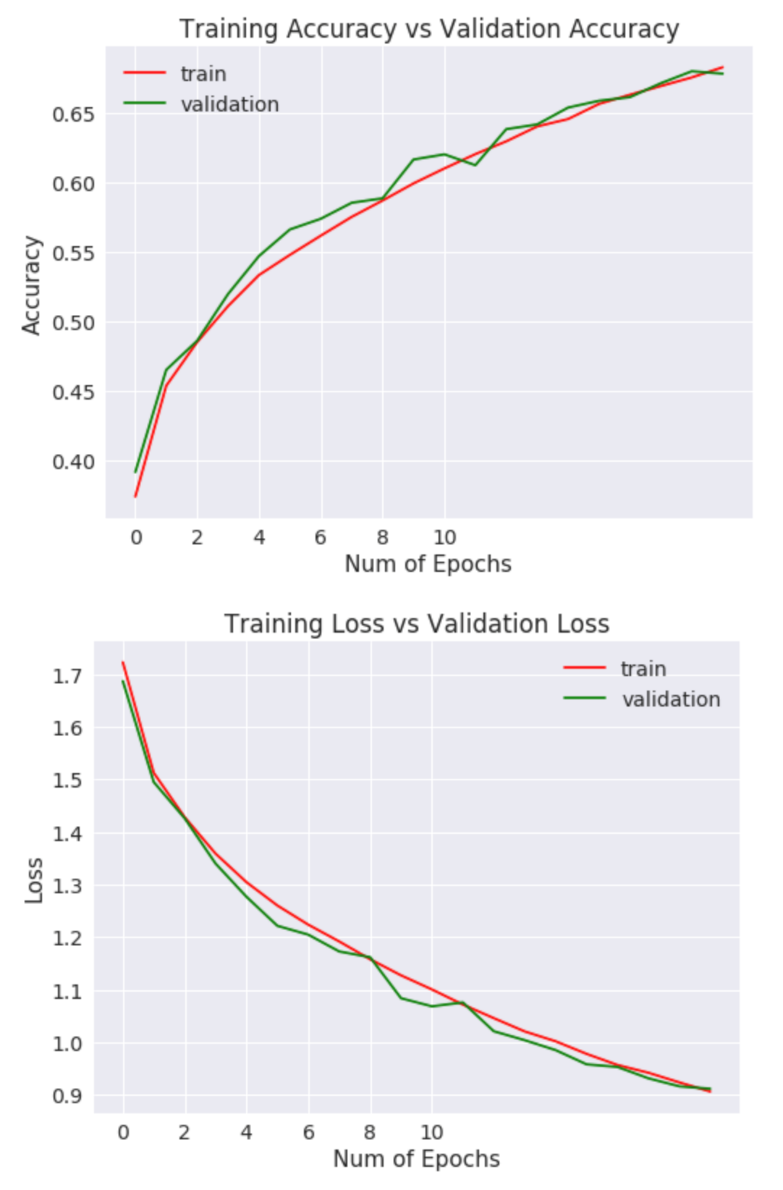
### **L4MOM**



### **L4MOM (0.25)**



### **L4MOM (fraction = 0.1)**



### **L4MOM (fraction = 0.2)**

