<u>EE4305 INTRODUCTION TO FUZZY/NEURAL SYSTEMS</u>

YEAR 2018/2019 (SEMESTER 1)

Individual Project

Submitted by: Tran Duy Anh

Matriculation Number: A0144185E

School of Electrical & Electronic Engineering

November 9th, 2018

Table of Contents

1 Literature Survey

CIFAR-10 is a popular dataset that has been experimented with many different machine learning algorithms. In this section, I will review 3 architectures that have performed relatively well on this dataset.

1.1 AutoAugment: Learning Augmentation Policies from Data [1]

1.1.1 Key features

There are many data augmentation operations in image processing. These include: translation, rotation, invert color, etc. This state-of-the-art architecture uses reinforcement learning to find the best data augmentation technique for each dataset. There are two numerical values associated with each augmentation policy, one is the probability of carrying out the technique /operation, the other is the magnitude to which it will be carried out.

There are 16 operations in total, all of which are from a python library called pillow[1]. Each operation comes with a magnitude and a probability which are discretized into 10 and 11 numerical values respectively.

Reinforcement Learning is then applied as a search algorithm. It comprises of two main components: a recurrent-neural-network controller and a proximal policy optimization algorithm. The reward signal used is how much improvement the policy has done on the validation accuracy during training of the mini-model.

In a nutshell, rather than fixing data augmentation process and tuning the neural networks hyper parameter, this paper fixes the neural networks and focuses on finding the best data augmentation process using reinforcement learning.

1.1.2 Advantages

When the best data augmentation technique is found on a dataset, it cannot be applied as effectively on other datasets. The most significant advantage of this architecture is, therefore, the transferability of the learned technique onto different datasets.

1.1.3 Disadvantages

One of the most significant disadvantages of this architecture is the computational complexity. There are 2.9×10^{32} possibilities within the search space for data augmentation policies.

1.1.4 Application area of image recognition algorithms

This work targets the beginning stage of a machine learning process, which is data manipulation/ preprocessing.

[1] Retrieved from https://pillow.readthedocs.io/en/5.3.x/ on November 8th , 2018.

1.2 Striving for Complicity: The All Convolutional Net [2]

1.2.1 Key features

This state-of the art architecture slightly modifies the conventional ConvNets by replacing the max-pool layer with an additional convolutional layer. Along with a pure convolutional layer approach, it also introduces some dimensionality reduction technique by using a stride of 2 instead of the conventional 1.

1.2.2 Advantages

In terms of performance, this method of replacing a max-pooling layer with a convolutional layer did perform better in CIFAR10 and CIFAR100. However, this might not hold true for other datasets.

1.2.3 Disadvantages

In terms of computational cost, using an additional convolutional layer requires more operations to be computed, and therefore, takes more time.

1.2.4 Application area of image recognition algorithms

This piece of work introduces a fresh approach to the conventional ConvNets. Understanding and changing up the conventional ConvNets structures might result in unexpectedly good performance.

1.3 Deep Networks with Internal Selective Attention through Feedback Connections [3]

1.3.1 Key features

ConvNets does not change its filter and weights after the training is done. This architecture challenges the conventional ConvNets' stationary and feedforward characteristic. It proposes a dasNet (Deep Attention Selective Network) architecture that simulates what a human brain does. DasNet can modify a convolutional layer's focus on its filter dynamically post-training. It employs Reinforcement Learning to modify the effect of each filter within the ConvNets.

1.3.2 Advantages

DasNet outperforms the traditional ConvNets in CIFAR10 and CIFAR100 datasets.

1.3.3 Disadvantages

Applying an additional Reinforcement Learning architecture during the classification results in some computational cost. Therefore, this architecture might not be applicable to tasks that require instant classification results.

1.3.4 Application area of image recognition algorithms

This piece of work focuses on the post-training stages of a ConvNets. Dynamically changing weights during classifications is certainly interesting and worth looking into.

2 Design an MLP classifier

2.1 Fine tuning:

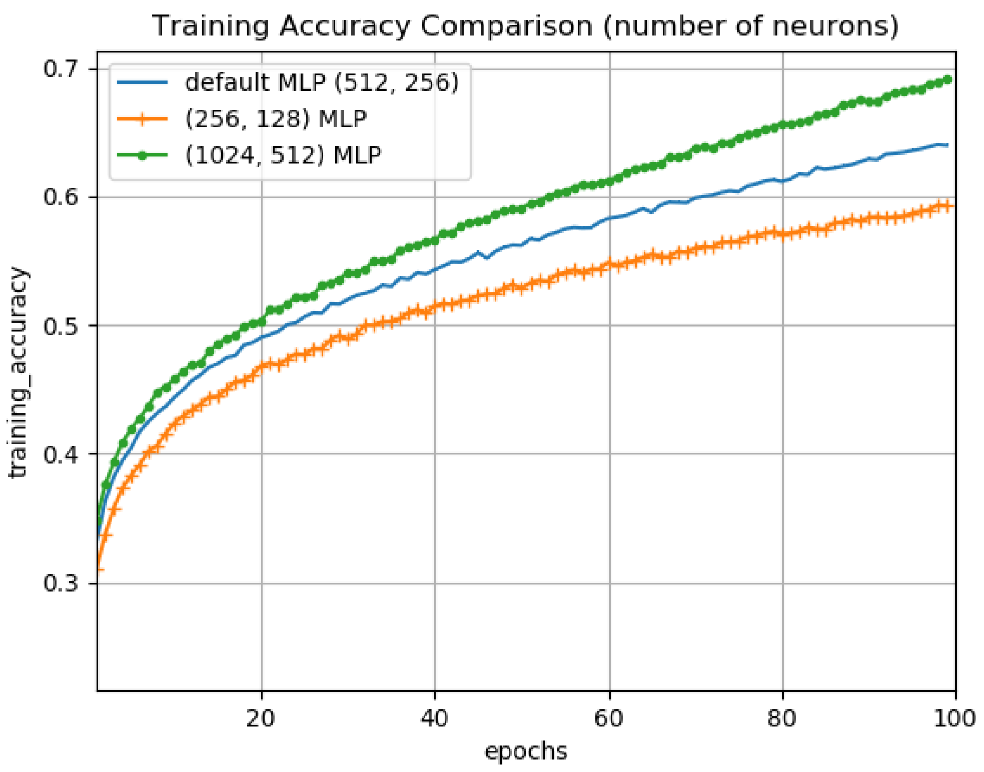
In this section, I modified 5 different parameters in the default MLP classifier and compared the performance (training accuracy and validation accuracy) of the modified models with that of the default one provided in the sample jupyter notebook.

2.1.1 Number of neurons in each layer

In the default MLP classifier, there are 4 layers in total:

- The input layers with 3072[2] neurons
- The first hidden layer with 512 neurons
- The second hidden layer with 256 neurons
- The output layers with 10[3] neurons

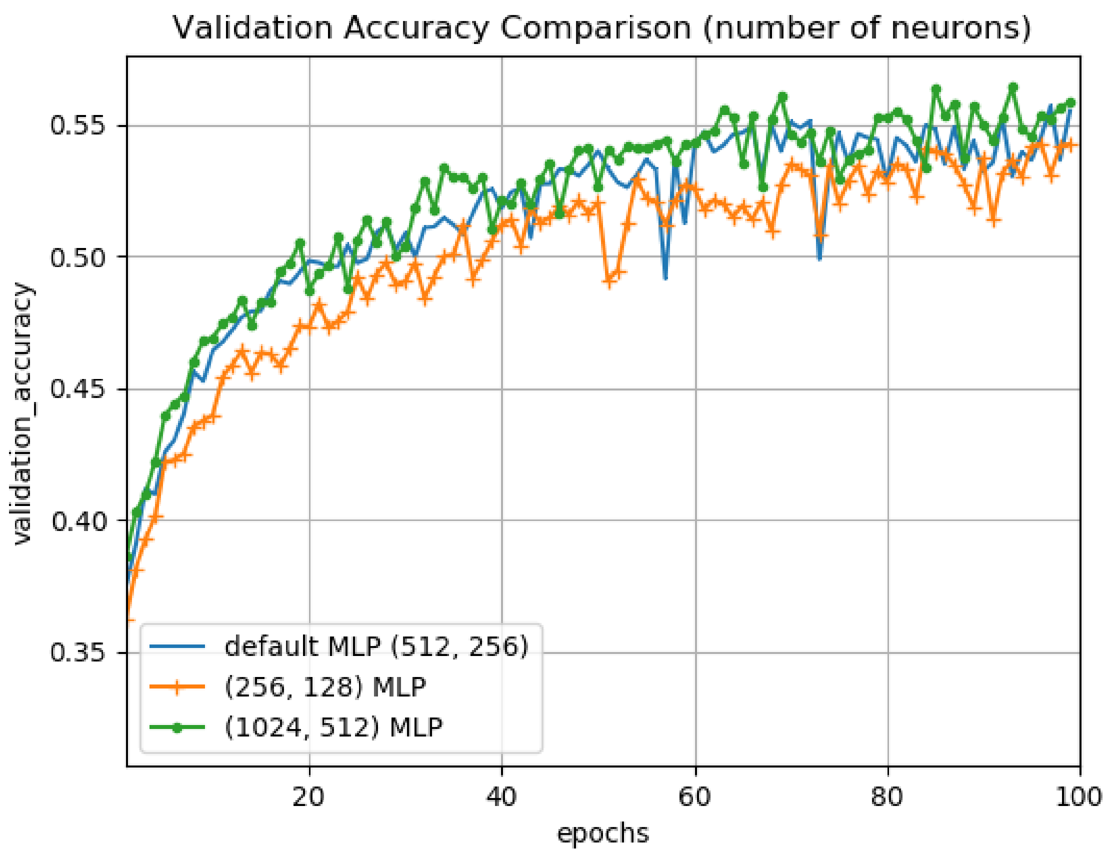
Since the number of neurons in the input and output layers are fixed, I changed the number of neurons in the two hidden layers[4].



[2] Each image in the CIFAR10 dataset is of size 32 pixels x 32 pixels. Each pixel contains 3 values representing red, blue and green. Therefore, 32 x 32 x 3 = 3072 neurons in input layer.

[3] There are 10 classes in the CIFAR10 dataset, hence the output layer must contain 10 neurons.

[4] (num1, num2) means the first hidden layer contains num1 number of neurons, the second hidden layer contains num2 number of neurons.



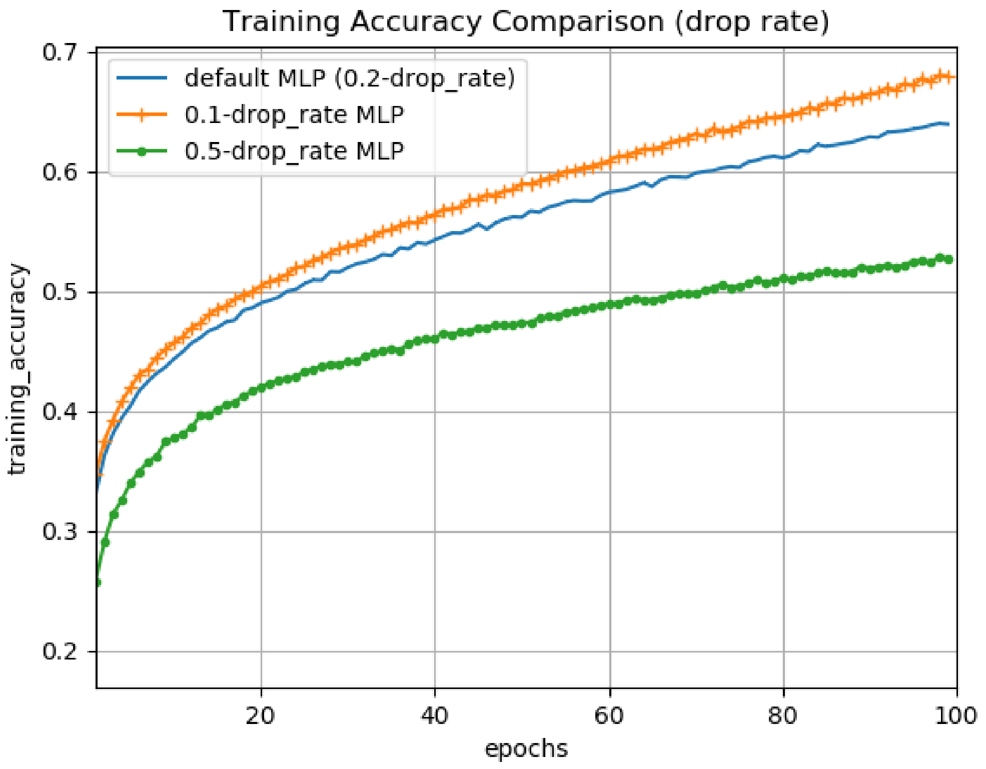
From the two figures, the training accuracy of the (1024, 512) MLP is better than the default MLP classifier, which means that a greater number of neurons in the hidden layers increases the training accuracy. The performance, assessed through validation accuracy, is not significantly different across the three cases. It is also worthy to note that during training, a greater number of neurons in the hidden layers lead to a much longer training time within each epoch.

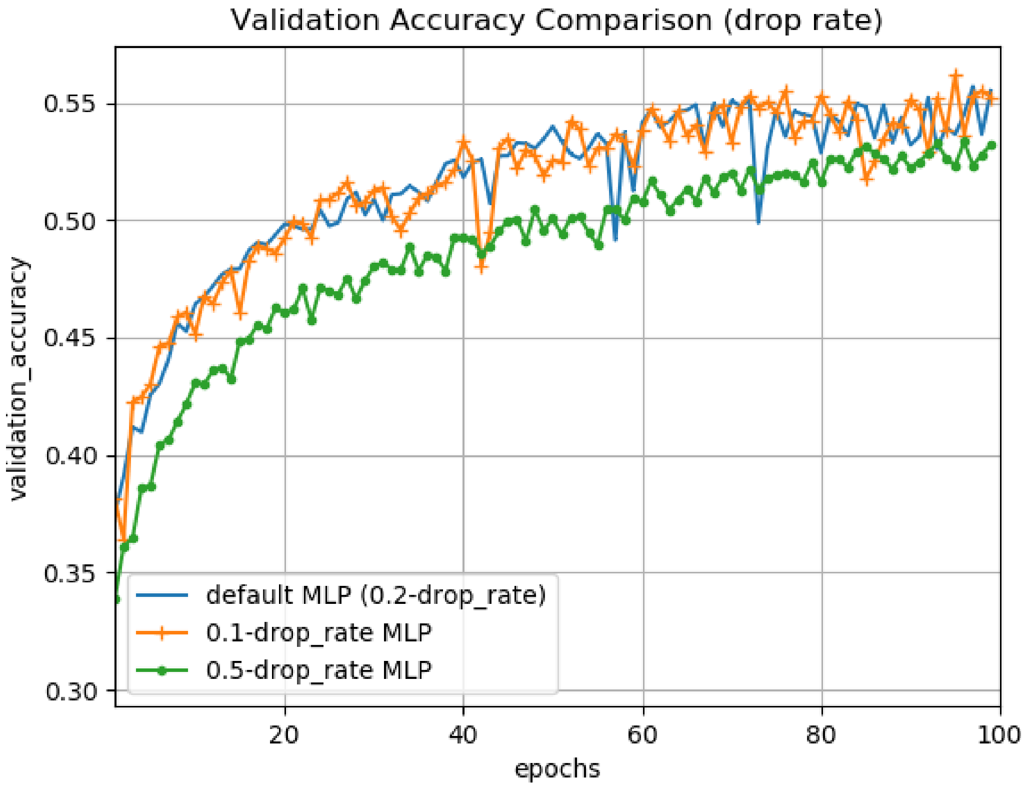
2.1.2 Drop-out rate

Drop-out is a method to reduce overfitting in a fully connected neural networks by randomly setting a fraction of input units to 0 [4][5].

In the default MLP classifier, the dropout rate is set to 0.2. I modified it to 0.1 and 0.5 and compare the performance.

[5] Retrieved from https://keras.io/layers/core/#dropout on November 8th , 2018.

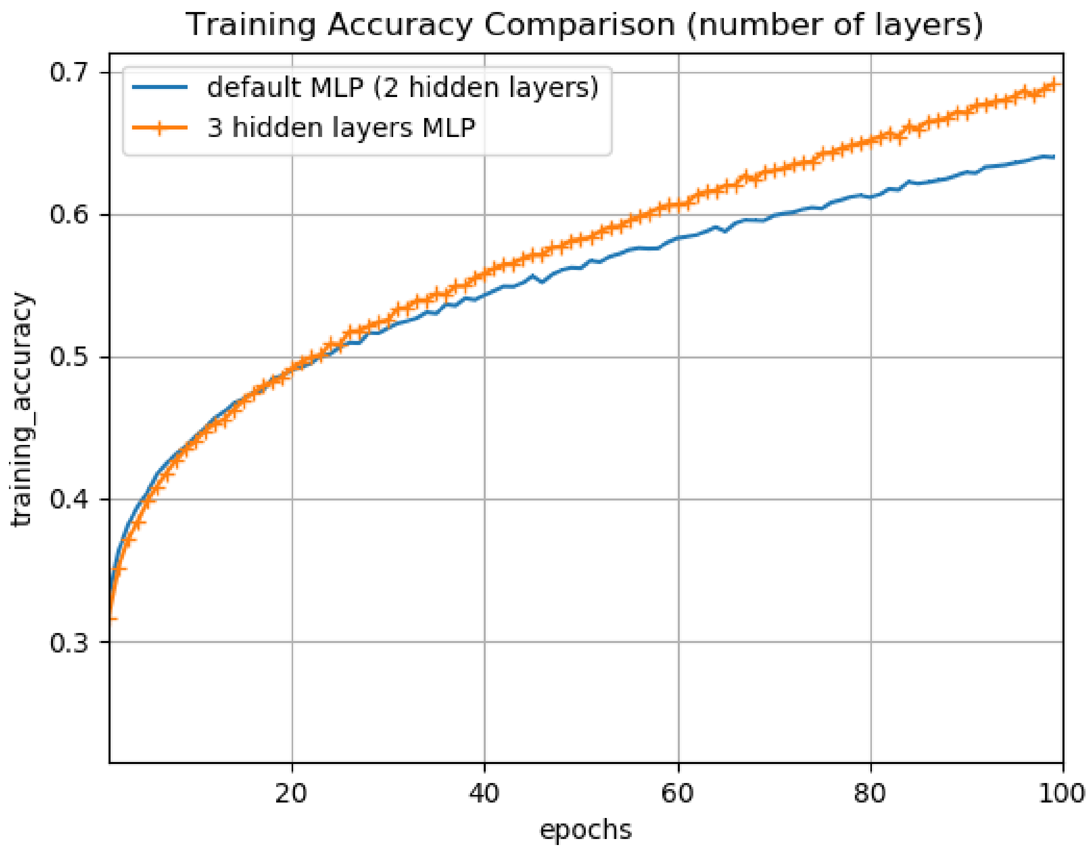

Training Accuracy Comparison (drop rate)
default MLP (0.2-drop_rate)
0.1-drop_rate MLP
0.5-drop_rate MLP
training_accuracy
epochs
0.2
0.3
0.4
0.5
0.6
0.7
20
40
60
80
100


Validation Accuracy Comparison (drop rate)
default MLP (0.2-drop_rate)
0.1-drop_rate MLP
0.5-drop_rate MLP
validation_accuracy
epochs
0.30
0.35
0.40
0.45
0.50
0.55
20
40
60
80
100

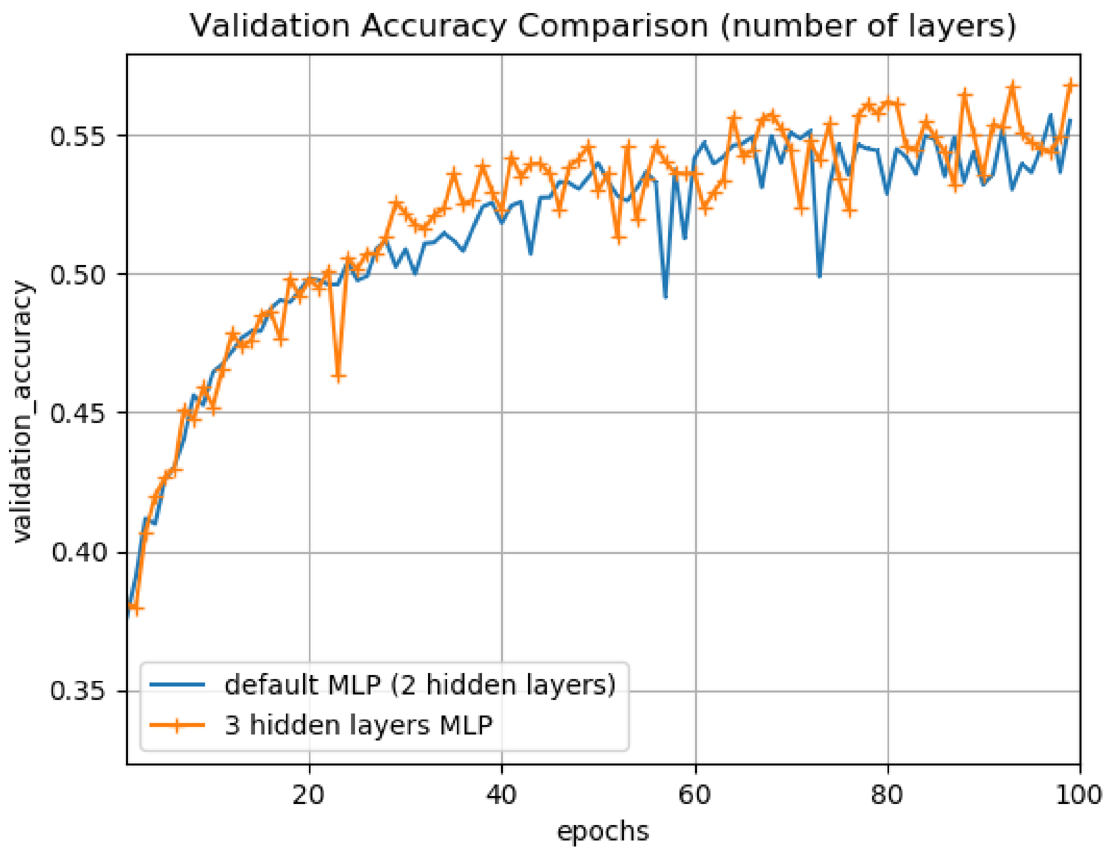
From the figures, a drop-out rate of 0.1 results in the highest training accuracy. In terms of validation accuracy, a classifier with 0.1 drop-out rate performs equally well as the default classifier. The one with 0.5 drop-out rate underperforms by a large margin.

2.1.3 Number of layers

As mentioned above, there are 4 layers in the default MLP classifier. I added one more fully connected layer with 1024 neurons between the input layer and the 512-neuron layer and documented the change.



From the figure, an additional hidden layer increases the training accuracy of the MLP classifier.



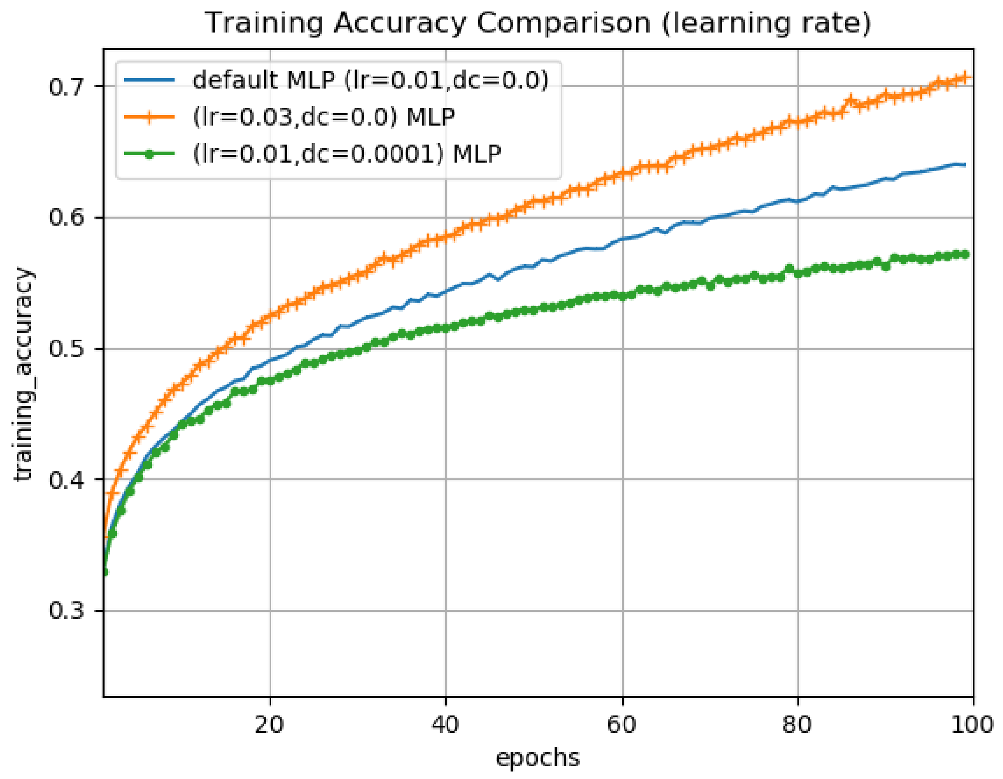
On the other hand, in terms of performance, the additional layer does not contribute significantly. The additional layer did make the training time significantly longer.

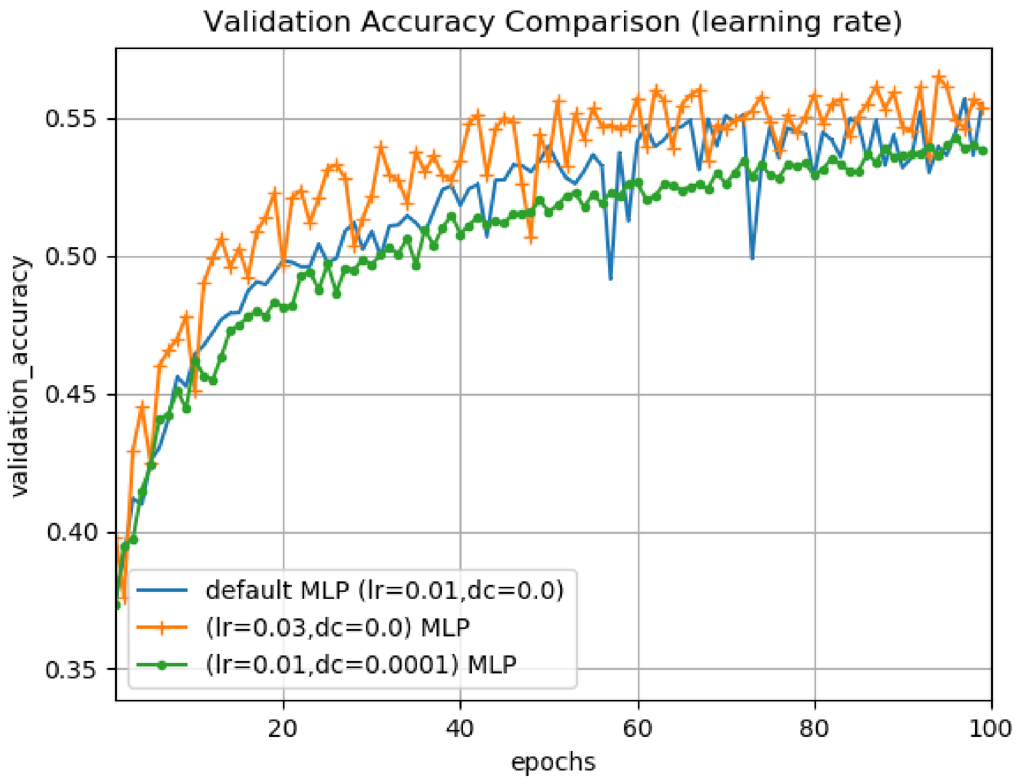
2.1.4 Learning rate

The learning rate decides how much the weights are modified after each iteration using stochastic gradient descent optimization technique [5]. The weight decay decides how the weights are regularized [6].

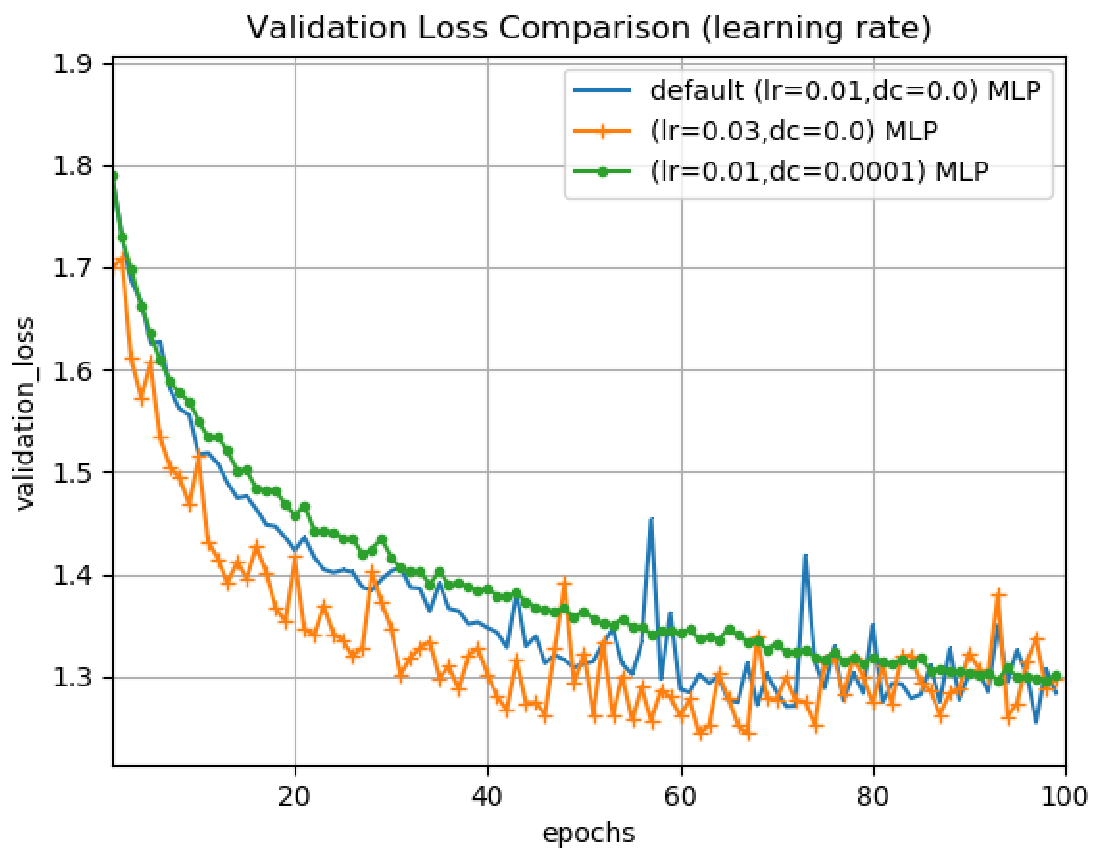
In the default classifier, the learning rate in the provided koras library's Stochastic Gradient Descent Optimization is 0.01 with decay 0.0[6]. I introduced two other classifiers, one with a learning rate of 0.03 and decay of 0.0 and one with a similar learning rate of 0.01 but with a decay of 0.0001.

[6] Retrieved from https://keras.io/optimizers/#sgd on November 8th , 2018.


Training Accuracy Comparison (learning rate)
default MLP (lr=0.01,dc=0.0)
(lr=0.03,dc=0.0) MLP
(lr=0.01,dc=0.0001) MLP
training_accuracy
0.7
0.6
0.5
0.4
0.3
20
40
60
80
100
epochs


Validation Accuracy Comparison (learning rate)
validation_accuracy
0.55
0.50
0.45
0.40
0.35
default MLP (lr=0.01,dc=0.0)
(lr=0.03,dc=0.0) MLP
(lr=0.01,dc=0.0001) MLP
20
40
60
80
100
epochs

As observed in the above figures, the larger learning rate results in an overall increase in training accuracy. Additionally, the validation accuracy of a learning rate of 0.03 did reach the peak performance much faster as compared to the other two.

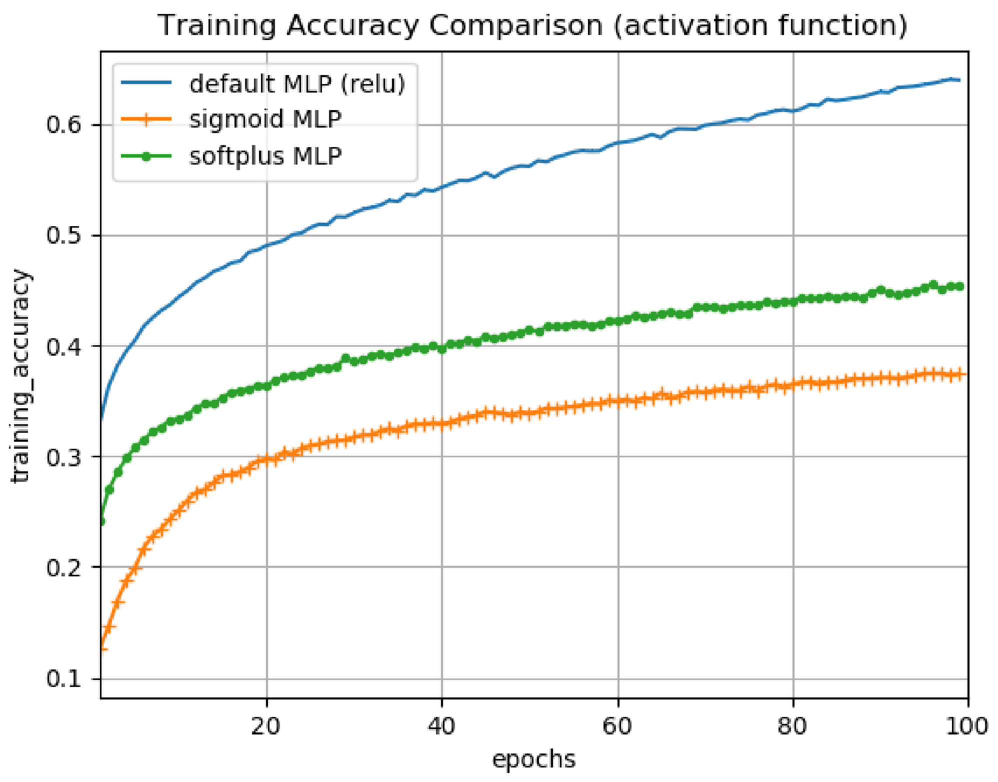


An additional metric, which is validation loss, is taken into consideration. It can be observed that higher learning rate leads to a faster decrease in the validation loss, meaning that the model converges faster to a local minimum.

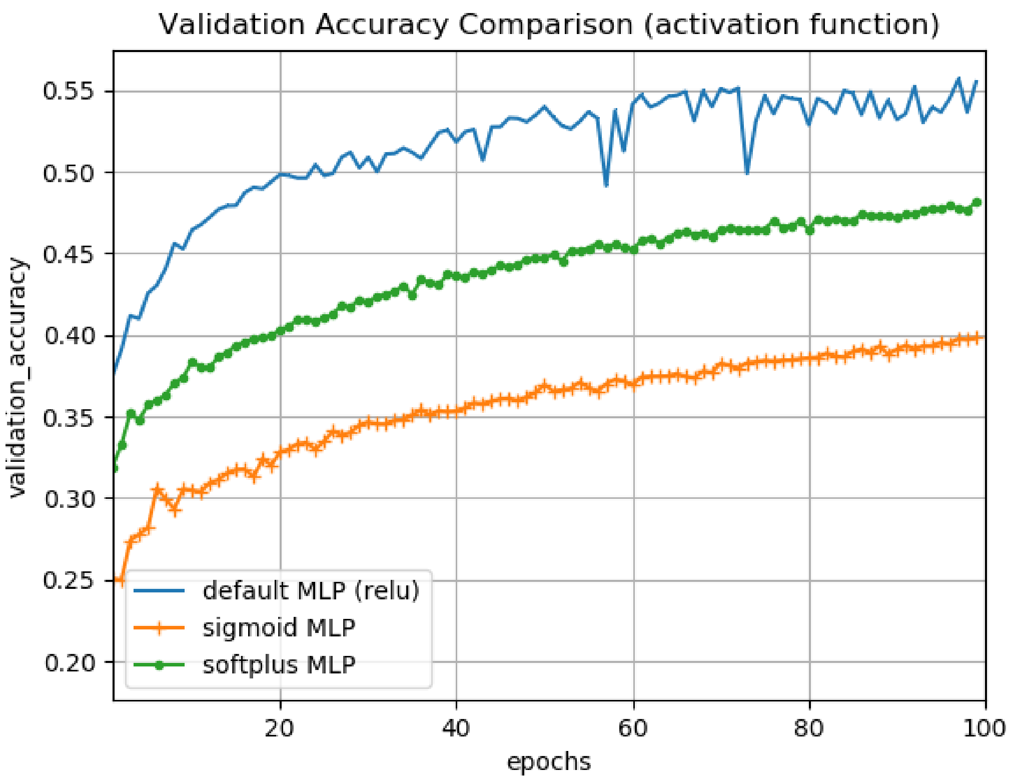
2.1.5 Activation function

There are many different activation functions in a neural network. However, Rectified Linear Units (ReLU) is proven the most suitable for CIFAR10 dataset due to its popularity among previous works. ReLU comes with the default MLP classifier in the sample notebook

Regardless, I tested the MLP classifier with two other activation functions: Sigmoid [5] and Softplus [6] to see how much of an effect activation function has on the overall performance.



Training Accuracy Comparison (activation function)
default MLP (relu)
sigmoid MLP
softplus MLP
training_accuracy
0.1
0.2
0.3
0.4
0.5
0.6
20
40
60
80
100
epochs



Validation Accuracy Comparison (activation function)
default MLP (relu)
sigmoid MLP
softplus MLP
validation_accuracy
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
20
40
60
80
100
epochs

The difference in both training and validation accuracy is obvious with ReLU performing best and Softplus performing worst. This test confirms ReLU's superior performance over other activation functions.

2.1.6 Conclusion

Some parameters are fixed due to the nature of the CIFAR10 dataset. Two of which is the loss function, which is "Categorical Cross Entropy" and the activation function in the output layer which is "SoftMax". This is because CIFAR10 is a multi-label classification with 10 labels.

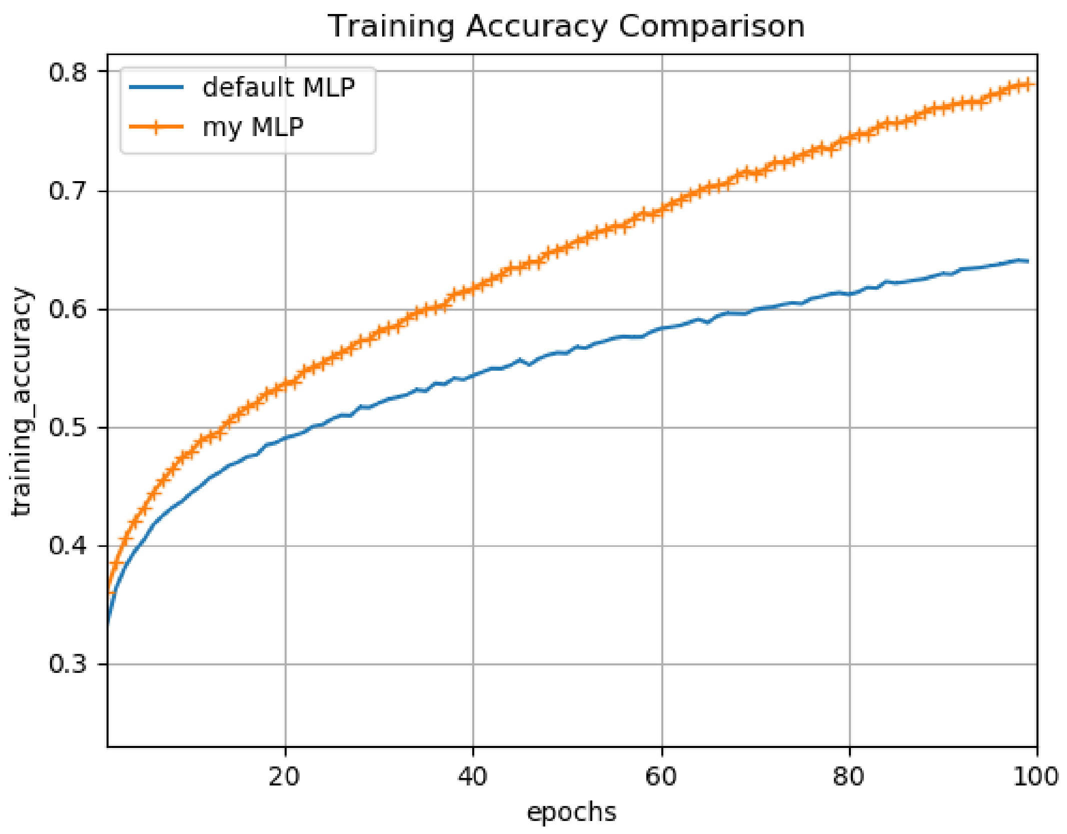
There are many other parameters that I can tweak to document their effects on the performance of the ml p classifier. These include different optimizers such as "Adagrad", "Adadelta", and "RMSprop". Due to time and resources constraint, I shall not address them in this report.

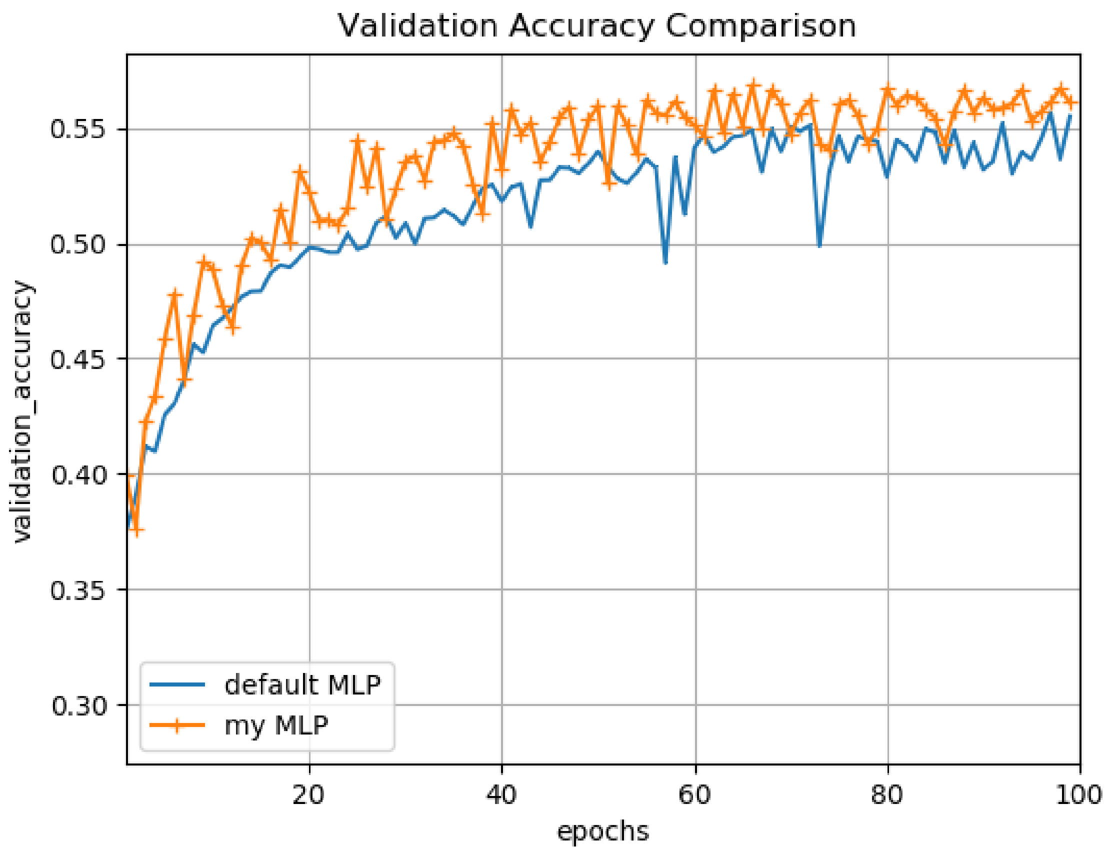
2.2 Design of MLP classifier

After the fine-tuning and testing, I designed my MLP classifier with the following parameters being modified:

1. **Number of layers/neurons:** my MLP classifier has 3 layers with 1024, 512 and 256 neurons respectively.
2. **Learning rate**: my ML classifier has a learning rate of 0.05.

2.2.1 Performance





2.2.2 Evaluation

1. **Strength:**

The two above figures indicated a superior performance of my MLP classifier to the default one. Mine outperforms in both training accuracy and validation accuracy. The speed of convergence is also faster due to higher learning rate.

2. **Limitations**:

The training time of my classifier is slightly longer. This has been addressed in section 2.1.3.

3. **Future work:**

As mentioned in 2.1.6, there are many other parameters that I could modify to achieve better results with an MLP architecture.

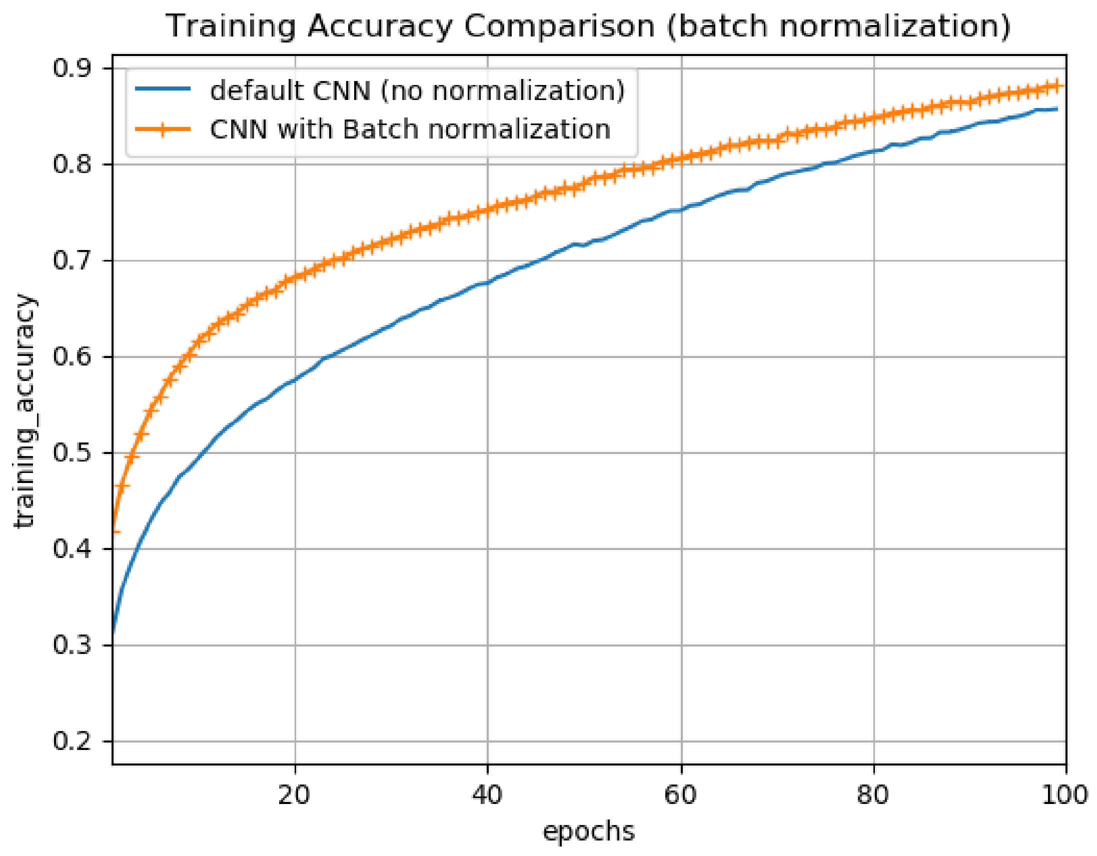
3 Experimenting with Deep Neural Networks (DNN)

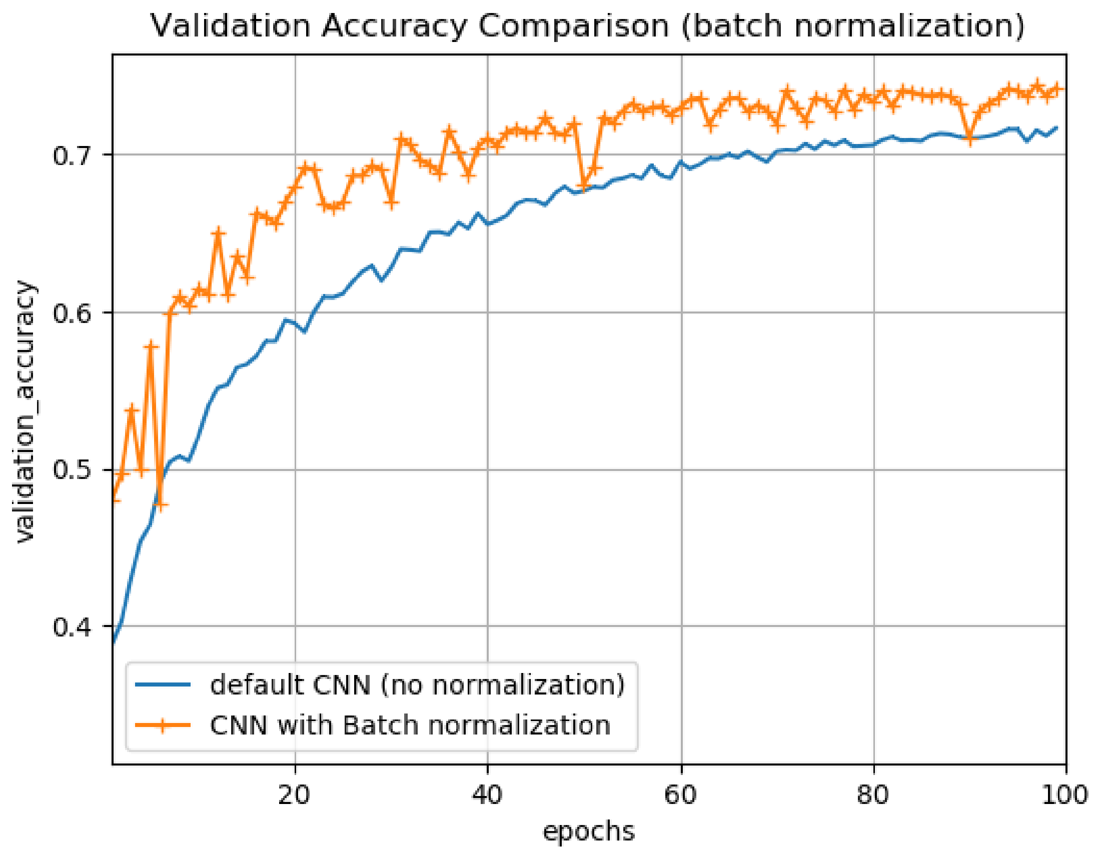
The training time of a DNN is significantly longer than an MLP classifier. Therefore, I only attempted a few modifications on DNN.

3.1 Batch normalization

Batch normalization is a method of shifting the input values of each layer to a range typically within 0 and 1. This ensure that no input has too high a value compared to other inputs, which may lead to the exploding gradient issue [7].

Since batch normalization modifies inputs, I applied batch normalization before an activation function in the fully connected layer and document the change in performance.

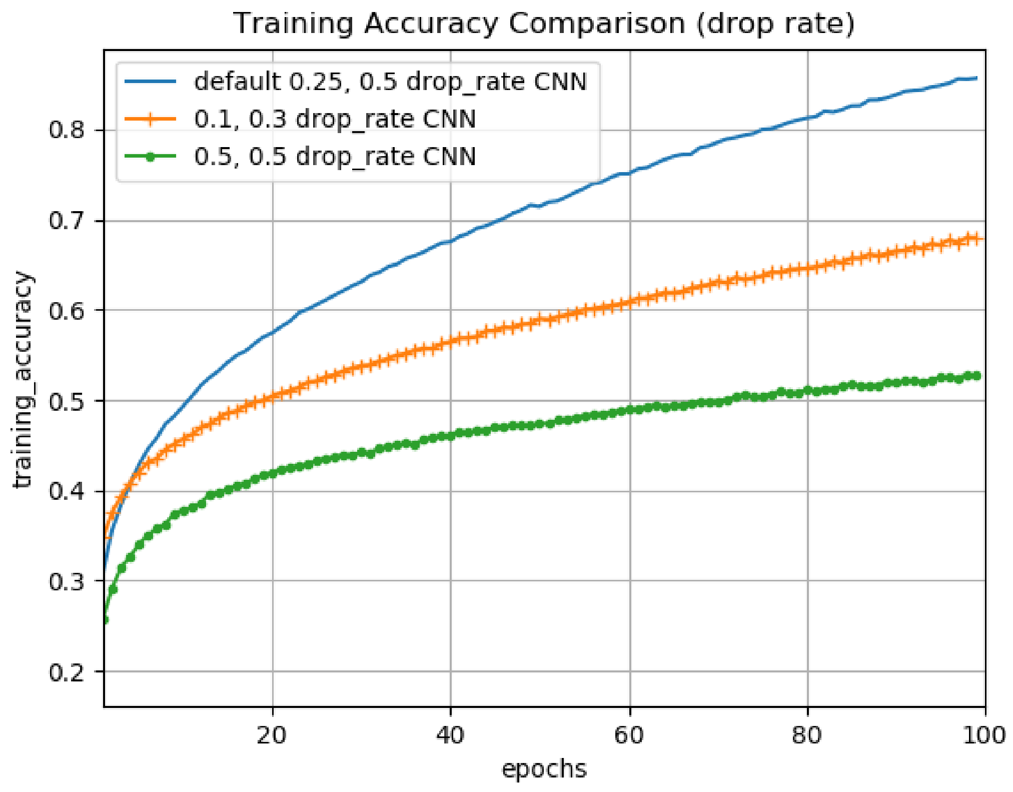


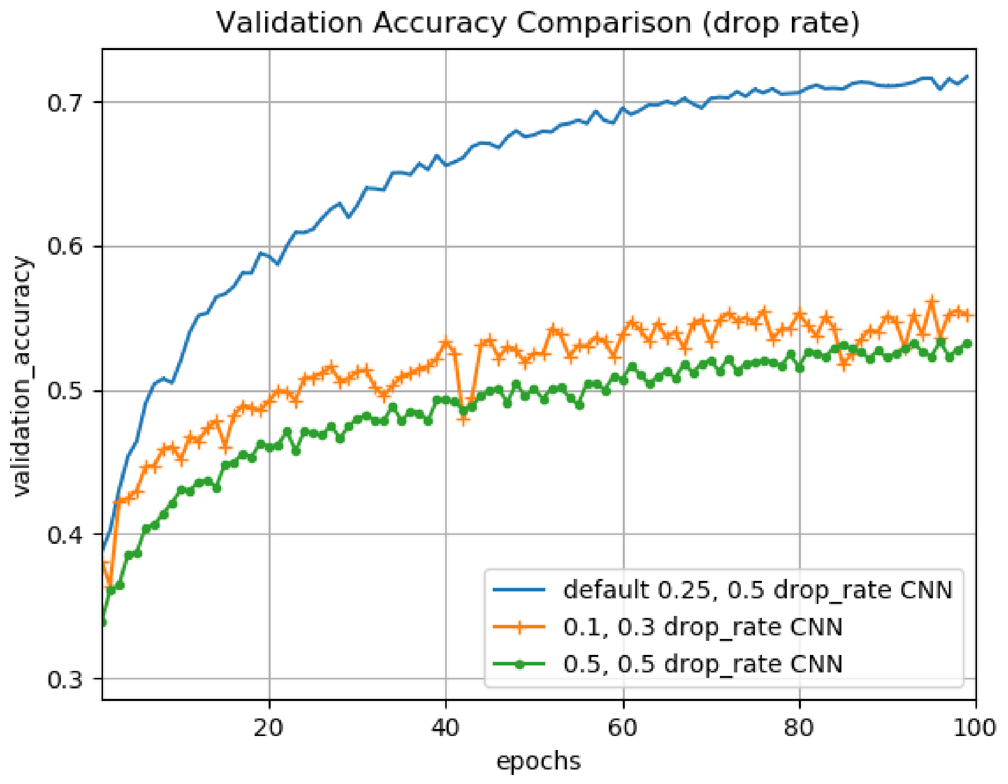


Batch normalization improvs the ConvNets performance in both training accuracy and validation accuracy without significant increase in training time. Therefore, batch normalization is recommended for ConvNets. It is also worthy to note that towards the end, the two curves are getting closer to each other. It might mean that if trained on more than 100 epochs, the default MLP can catch up.

3.2 Drop-out rate

In the default ConvNets classifier, there are 2 drop-out rates of values 0.25 and 0.5 respectively. For this reason, I created two classifiers, one with a 0.1 and 0.3 drop-out rates, and one with a 0.5 and 0.5 drop-out rates and assessed the effect of drop-out rates on ConvNets classifiers' performance.

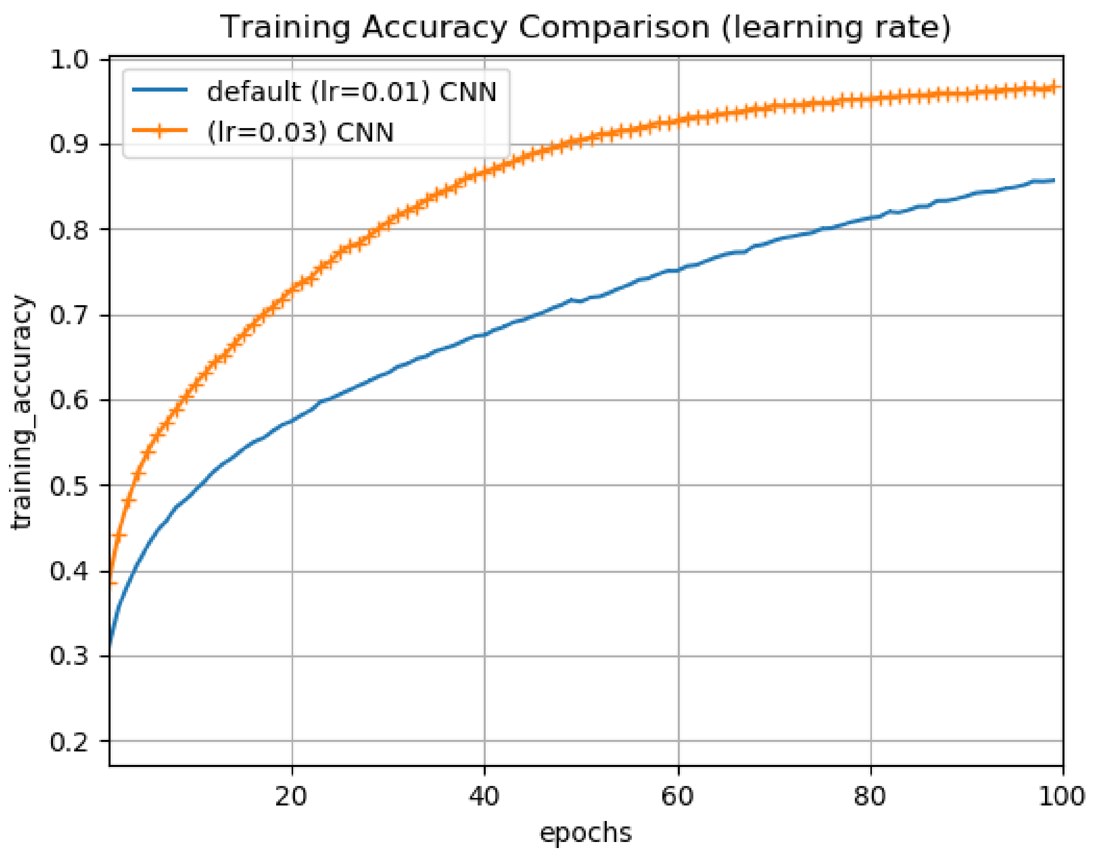

Training Accuracy Comparison (drop rate)
default 0.25, 0.5 drop_rate CNN
0.1, 0.3 drop_rate CNN
0.5, 0.5 drop_rate CNN
training_accuracy
epochs
0.2
0.3
0.4
0.5
0.6
0.7
0.8
20
40
60
80
100

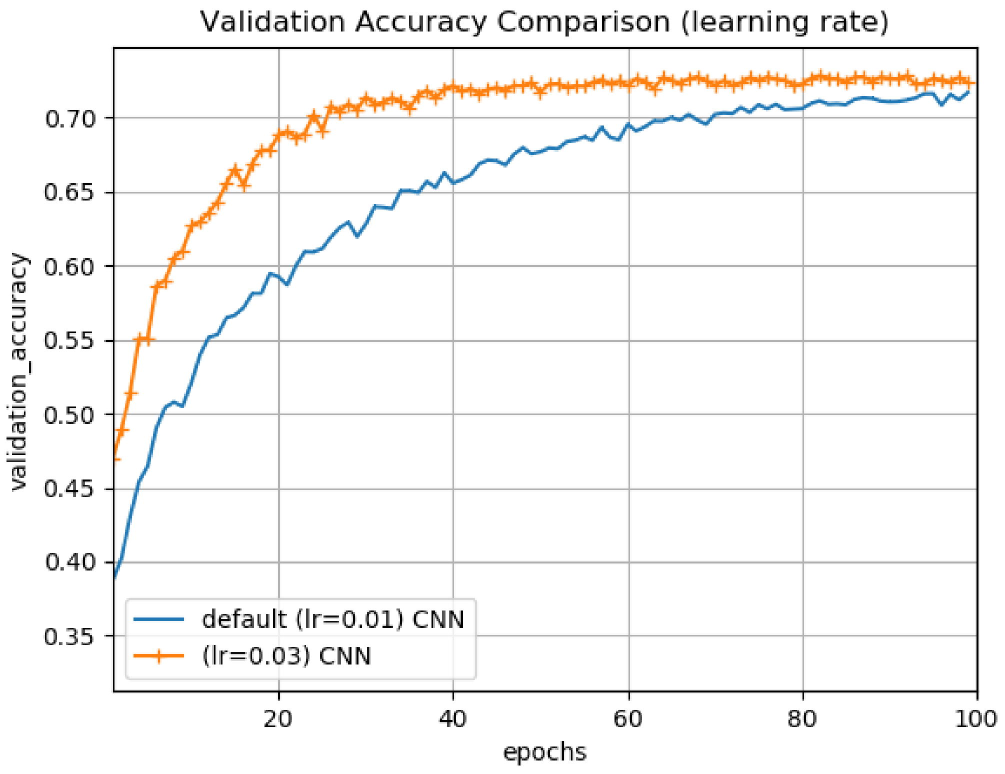

Validation Accuracy Comparison (drop rate)
default 0.25, 0.5 drop_rate CNN
0.1, 0.3 drop_rate CNN
0.5, 0.5 drop_rate CNN
validation_accuracy
epochs
0.3
0.4
0.5
0.6
0.7
20
40
60
80
100

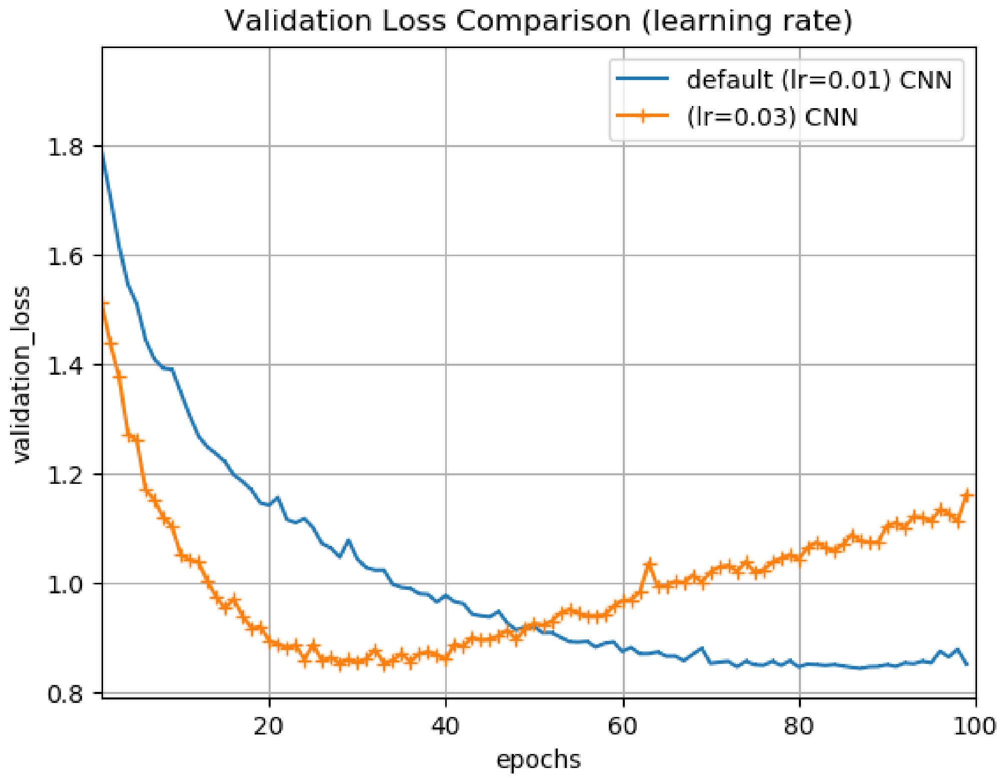
The default ConvNets classifier outperforms the other two by a large margin. Given more time and resources, I would like to explore more variations of drop-out rates and figure out different combinations that can result in better performance.

3.3 Learning rate

Like section 2.1.4, the default ConvNets classifier uses a learning rate of 0.01. I modified it to 0.03 and recorded the change in performance through each epoch.




Validation Accuracy Comparison (learning rate)
validation_accuracy
epochs
default (lr=0.01) CNN
(lr=0.03) CNN
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
20
40
60
80
100


Validation Loss Comparison (learning rate)
validation_loss
epochs
default (lr=0.01) CNN
(lr=0.03) CNN
0.8
1.0
1.2
1.4
1.6
1.8
20
40
60
80
100

A higher learning rate results in a higher training accuracy and validation accuracy. Like part 2.1.4, the ConvNets with higher learning rate manages to reach 0.72 (the highest validation accuracy) faster than the default ConvNets. One interesting observation is about the validation loss. The ConvNets with a higher learning rate reaches the bottom much earlier (at around the 30th epoch) but then steadily increases towards the end, while the default ConvNets keeps on decreasing.

3.4 Conclusion/ Evaluation

ConvNets is a much more complicated architecture as compared to a Multi-Level perceptron architecture. Despite their huge computational cost, models built from ConvNets outperforms a simple MLP model by a large margin. Given more time and resources, I would like to explore different DNN structures like Residual Networks (ResNets) and different data augmentation methods as mentioned in 1.1. The use of Reinforcement Learning will also be investigated to improve the performance of DNN before, during and after training. Expanding my knowledge in Deep learning will also allow me to construct a DNN models from scratch and construct different layers to optimize my networks. Overall, this project was a bit of a learning curve for me, and I enjoyed it very much.

4 References:

[1] Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V. and Le, Q.V., 2018. AutoAugment: Learning Augmentation Policies from Data. *arXiv preprint arXiv:1805.09501*.

[2] Springenberg, J.T., Dosovitskiy, A., Brox, T. and Riedmiller, M., 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

[3] Stollenga, M.F., Masci, J., Gomez, F. and Schmidhuber, J., 2014. Deep networks with internal selective attention through feedback connections. In *Advances in neural information processing systems* (pp. 3545-3553).

[4] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), pp.1929-1958.

[5] Jacobs, R.A., 1988. Increased rates of convergence through learning rate adaptation. *Neural networks*, *1*(4), pp.295-307.

[6] Krogh, A. and Hertz, J.A., 1992. A simple weight decay can improve generalization. In *Advances in neural information processing systems* (pp. 950-957).

[7] Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.