Project-1

Experimental Results:

In this report, we cover extensive experimentation of the CNN architecture, specifically, the Y-network which is completed on the CIFAR10 and dataset CIFAR100 (single experiment only).

1.) CIFAR10 and CIFAR100 results and Network configuration:

i.) CIFAR10:

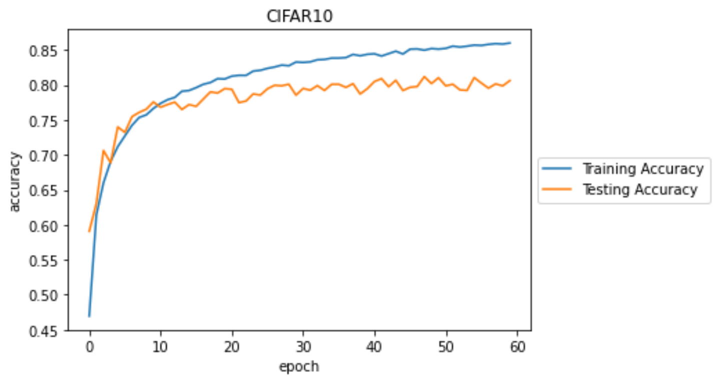
The performance for this dataset was measured using the test data provided in the mentioned dataset. We use the model as per the requirements having the following hidden layers: 3 convolutional layers with (maxpooling and dropout layers for these convolutional layers), 1 FC (dense) output layer, with all but the last transfer functions as 'ReLU' and the last one as 'Softmax'. We use 60 epochs, batch size of 64, a dropout of 0.1, and use the Adam optimizer. Finally, data augmentation was used in both sets to build a stronger and more robust model:

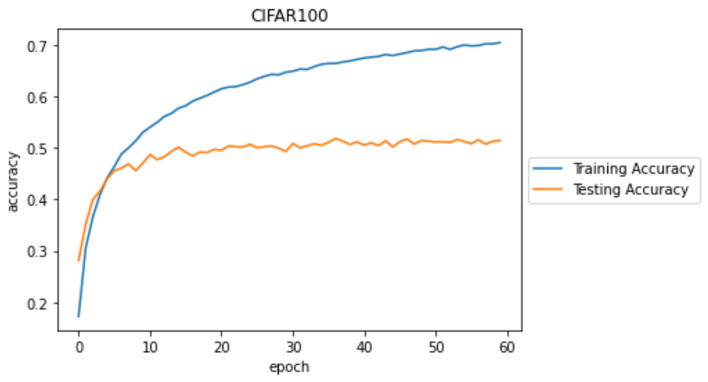
Final Training Error: 86.06%
Final Testing Error: 80.63%

ii.) CIFAR100:

For this we modify the network to be more complex to classify the 100 classes. Additionally, we use the same concept for transfer functions as given in i.) as well as the batch size, optimizers, epochs remain the same as well. The network configuration is given below.

Final Training Error: 71.02%
Final Testing Error: 51.49%





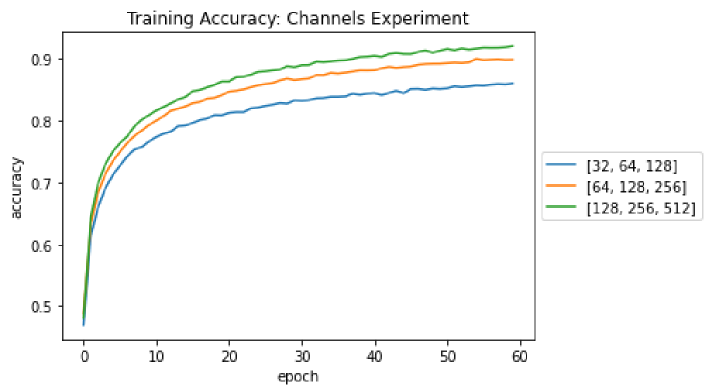
2.) Model Augmentation Experiments:

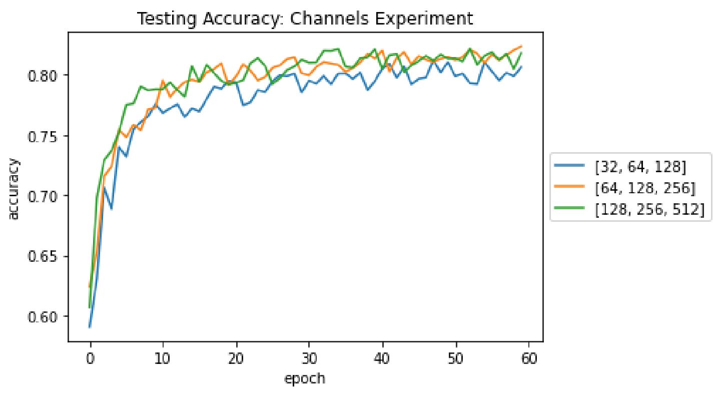
i.) Number of Channels:

Here, we consider 3 models with different (factor of 2) number of channels. The results show that the training errors are lowest for the most filter channels used which makes sense as we have more complexity to represent and potentially learn the data itself (overfitting) when we increase the number of channels. On the other hand, the model with [64, 128, and 256] channels (2nd rank in terms of channel sizes) performed the best on the validation set.

This shows that the model 2 is the best fit in response to the overfit v/s underfit problem i.e., the config [32, 64, 128] had an underfit issue and the one with [128, 256, 512] channels had an overfit issue yielding problems in either being unable to handle the data complexity (underfit) or overcompensating for the same (overfit).

The descriptions in square braces represent number of channels and the final number of output nodes. We obtain the below 2 graphs for accuracies measured on the training and testing data.

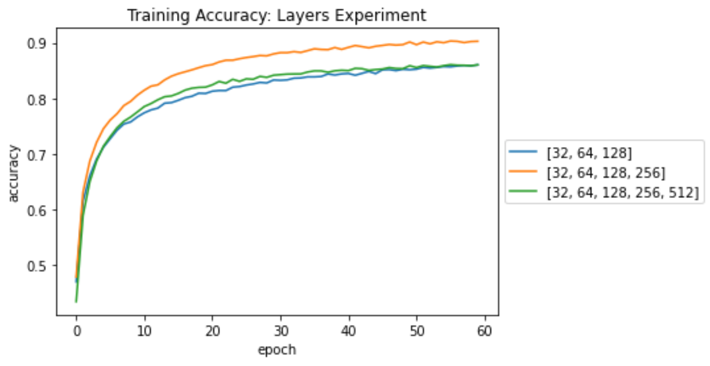


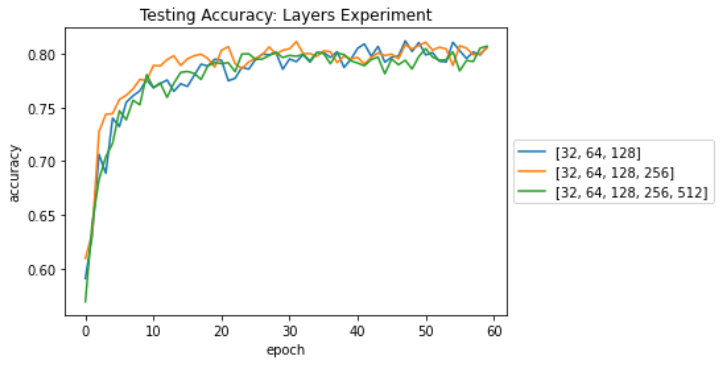


ii.) Number of Layers:

Now we use 3 different models with varying number of hidden convolutional layers [add 1 consecutively]. An odd behavior is observed in this case, we know that the increase in the number of layers should increase the training accuracy due to overfitting. However, the model with the greatest number of conv layers performs almost as good as the model with the least on the training set itself.

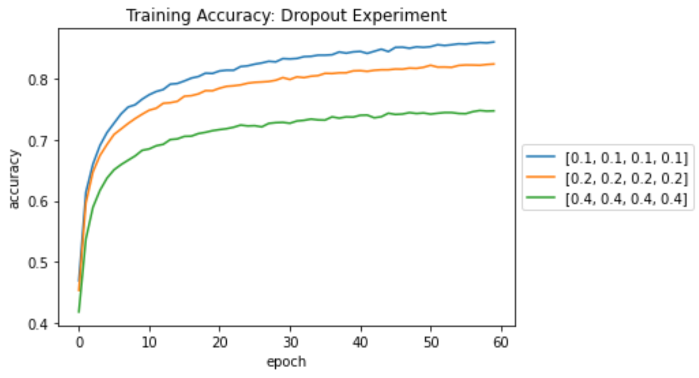
Upon further investigation, I observed that the subsequent addition of the maxpooling layers after every conv layers were affecting the model and proved to be a bottleneck in this case thus giving us a key observation that we shouldn't design models in a way that pinches off the flexibility added by the new layers by adding maxpooling layers as they'd repeatedly divide the image output into half hitting an all time low at the end section layers. Instead, fractional maxpooling must be achieved by setting stride as e.g., 2 but the window be>2.

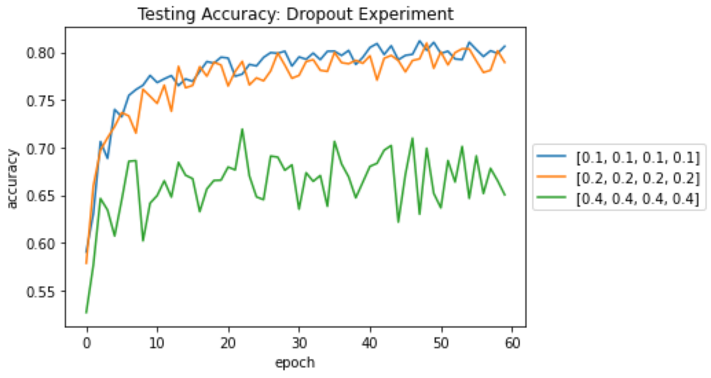




iii.) Dropout Configuration:

For the dropout configurations, we keep the base model same for all dropouts (like CIFAR10 model discussed in 1.)) We experiment dropout with ratios 0.1, 0.2, and 0.4 between all the conv and the last FC layers. The following results were obtained:

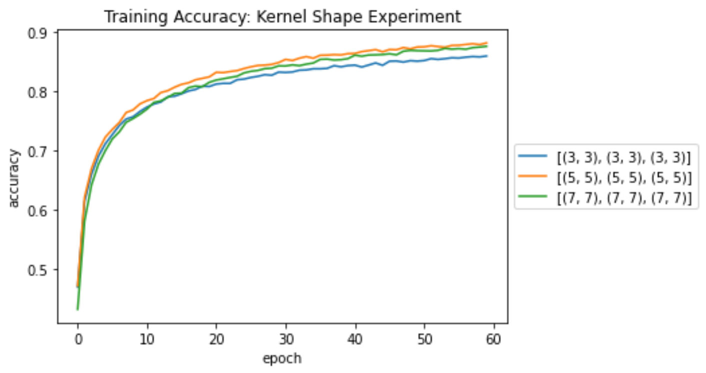


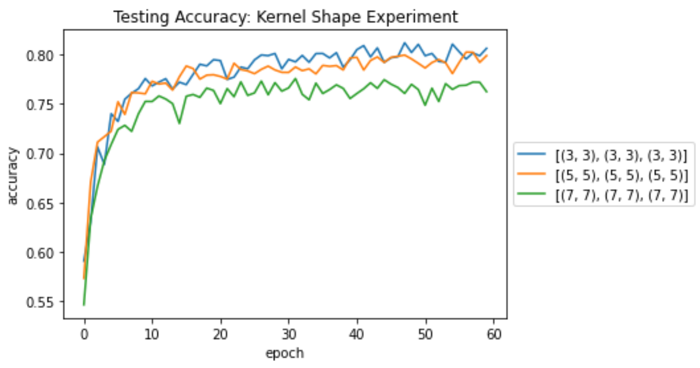


We observe that when dropout ratio is increased the training accuracy is reduced while the testing accuracy peaks at the ratio between 0.1. From the study of the number of channels we know that the base model with config [32, 64, 128] is underfitting already when compared to the model 2 with twice the number of channels. This indicates that any dropout layers will not be much useful as they push the model towards more underfitting which is destructive in this case, beyond 0.2 the model is visibly underfitting the dataset in this case.

iv.) Kernel (filter) Sizes:

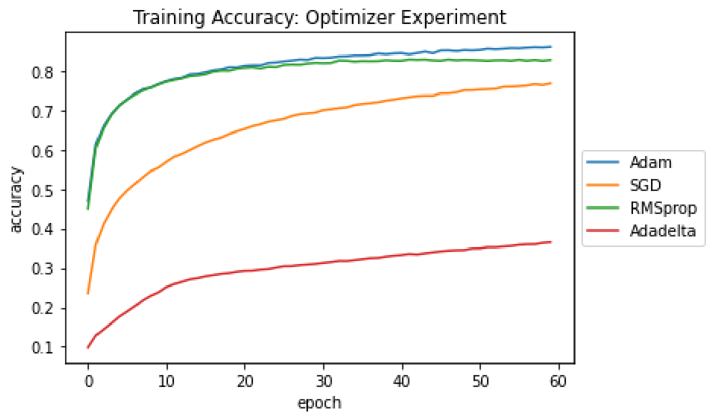
We augment the kernel sizes for all the conv layers to observe their impact on the accuracies. The 3 configurations used have sizes [(3, 3), (5, 5), (7, 7)]. We see that the kernels of size 5x5 and 7x7 don't have much difference, this is mostly because after 3 maxpooling layers the base output image is itself 4x4 (divide 32 by 2x2x2). Usually for higher resolution images wherein this bottleneck is not observed we can see the effects caused by high kernel sizes there, additionally, when the kernel is of smaller size, much more intricate details are learned and captured by the network (local features/ patterns) which can then be used by higher order layers to deduce these patterns further. In this case as well we see that the smaller size of 3x3 produces the best results on the testing set due to the reason mentioned before.

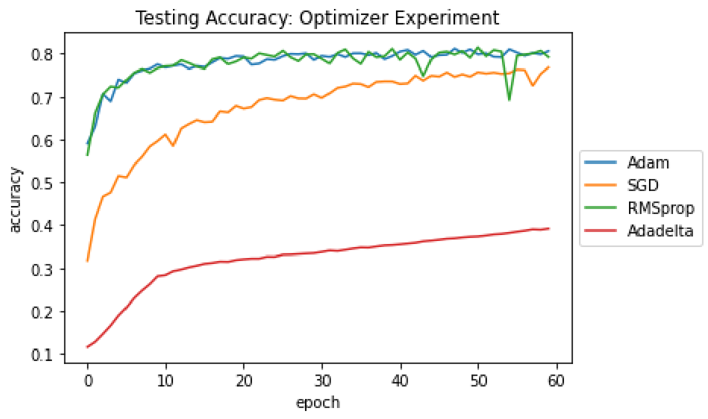




v.) <u>Optimizers:</u>

The optimizers used in this experiment are ‘Adam’, ‘SGD’, ‘RMSprop’, and ‘Adadelta’. The base model is the same as in 1.). The following graphs were obtained on training and testing accuracies with fixed number of epochs.

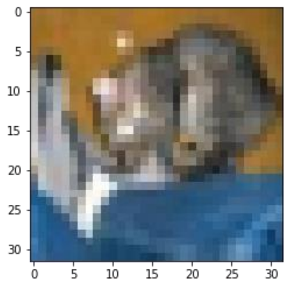




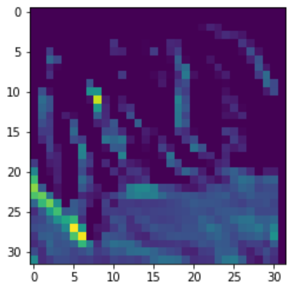
We can observe that for the training set ADAM seems to perform slightly better than RMSprop and largely better than Adadelta and SGD. This changes when we shift to the testing domain. On the testing graph, Adam performs almost as RMSProp. Additionally, Adam has a much more stable curve. ADAM overcomes the issue of better updates in the direction of the descent by using internally RMSProp and Momentum. Here, RMSProp takes care of oscillations in updates and performances by dividing the RMS for all the features rendering them less prone to high variations. Therefore RMSProp tapers off at the end section of epochs from ADAM in terms of accuracy because it doesn’t use the momentum factor in its updates making slower progress. Additionally, the SGD has no guarantees of convergence and may get stuck in a local minima saturating at a lower accuracy as it also doesn’t have the momentum factor. Finally Adadelta performed the worst updates which is in part due to adaptive learning rate based on the development under each dimension. This suggests that the adaptive rate would make it possible to learn even when other methods have saturated, however, the con is that the number of epoch need to be much higher to even match the performance of ADAM in our case.

3.) Feature map display:

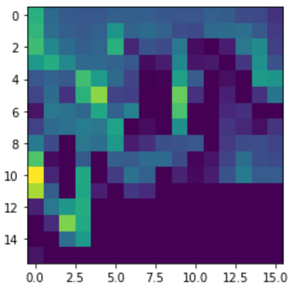
We have the feature maps for our main network configuration for CIFAR10. When we present them in a series as shown below, it can be observed how the initial layers might be performing regional low-level feature extraction, later on we see certain features activate certain sections of the output and those can correspond to some high-level features like texture (furry, plane, etc.) It is also interesting to observe that the initial maps suggest that the processing puts less weights on monotonic areas producing zero-outputs with no changes at all and focusses on some boundaries.



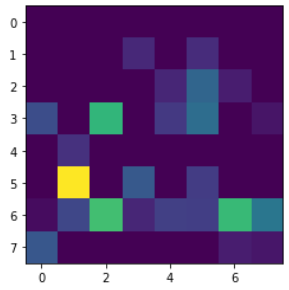
Input Image



1st Conv Layer Output



2nd Conv Layer Output



3rd Conv Layer Output

We can see that it's not completely random but has a patterns of edges in the first layer highlighted by light-blue color representing larger values, we also see the textures with some variations to produce an activation and monotonic patches with next to zero changes produce no or low output activation, the 2nd conv layer seems to be performing high-level associations leasing us to the final 3rd conv layer output which has the strongest activation for the location (1, 5) it may represent presence of a part very closely related to the a class e.g., an 'ear'. After these layers we perform another maxpooling operation and end with an FC dense layer. These final layers might be performing a weighted average on the feature maps to achieve something like this: 0.1 x (legs) + 0.5 x (roundhead) + … and indicating the probability of a person and so on.

Discussion:

The following are the key takeaways from this report and the experimentations performed.

- CNNs perform regional processing in the initial layers and high-level associations in the later layers.
- Maxpooling layers put a cap on learning by reducing the size of the output feature map, to combat this one must use window size > stride = 2 to add fractional size reduction.
- Higher number of channels increases the model complexity and a size of [64, 128, 256] performed best for our CIFAR10 dataset.
- Smaller kernel sizes in the initial layers should be used especially if the input image size is so small,
- Dropout layers will only be more helpful if the base model config is itself not an underfit, otherwise we are just increasing the same issue by using dropout layers reducing performance.
- ADAM performance is usually good, if you don't know the nature of the data, one can use this.