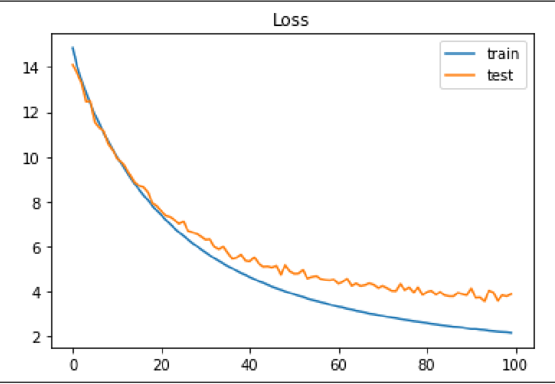
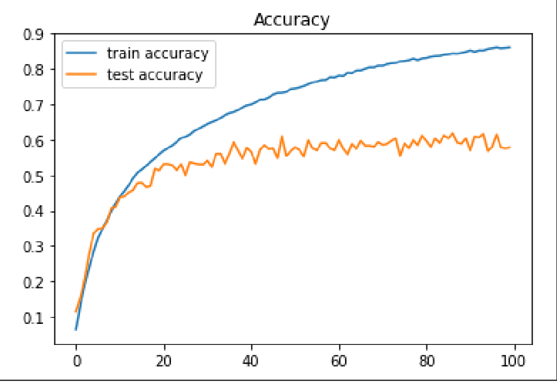
CSE 676 - Deep Learning
Project 1 Report

Charanteja Kalva
UB person: 50336873
UBIT name: charante

1 Introduction:

- This Project is designed with an objective to observe the effect of regularizations like Batch normalization and Dropout on various deep networks like VGGNet, ResNet and Inception V2. This project helps us to understand how different networks behave under different settings. We also explore the networks with different optimizers like SGD and ADAM.
- In my findings I observed that even a small change to the network settings can make the networks perform better in some cases and even perform bad in some cases, so we need to be careful and thoughtful before changing the parameters.
- In this project I have implemented a total of 18 variants of deep neural networks with different settings of VGG, Resnet and Inceptionv2 over CIFAR-100 dataset. Below table1 has the values for recall, precision and accuracy for all the variants. And I have performed a few more experiments by tweaking the parameters and by combining different regularizations.
- The Best model I was able to build was with VGG Net 16 with SGD optimizer, BatchNormalization, Dropout and L2 weight Regularization, which gave the following values.
  - **Accuracy: 61.85%**
  - **Recall: 61.85%**
  - **Precision: 63.75%**
- If we observe the below graphs of loss and accuracy, the model is perfect in terms of theory, i.e. the loss is steadily decreasing and accuracy is steadily increasing, there is no overfitting of training data and the model has low bias and low variance.

Loss Graph of VGGNet16(best achieved)	Accuracy Graph of VGGNet16(best achieved)
	

- I have enhanced only VGGNet to see how the model performs with all 3 regularizes. I am sure that we can also improve the results of ResNet and InceptionV2 by tweaking few parameters. We cannot justify on which model is good or bad cause every model has its pros and cons in its own setting.

- I have used Early stopping on Validation accuracy.
- Before observing the effect of regularizations like Batch normalization and Dropout, lets see briefly what they do.
- **BatchNormalization** during training, we normalize each layer's inputs by using the mean and variance of the values in the current mini-batch[1].
- Advantages of using BatchNormalization.
  - Networks train faster
  - Allows higher learning rates
  - Makes weights easier to initialize
  - Simplifies the creation of deeper networks
  - Provides a bit of regularization
- **Dropout** as the name implies some nodes in the network are ignored while doing forward and backward propagation with some probability. Main advantage of Dropout is to avoid Overfitting of Training data.
- I have normalized the input data before training the network in all the models. I have referred to the code snippet from [2]

2 Values for all variants of VGGNet 16, ResNet18 and InceptionV2.

	Architecture	VGGNet 16			ResNet 18			Inception V2		
optimizer	Variant	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy
SGD	BatchNorm	56.32	52.67	52.67	52.81	51.68	51.68	48.83	49.1	49.1
	Dropout	46.24	45.68	45.68	46.06	45.97	45.97	50.14	49.11	49.11
	No Regularization	39.60	39.87	39.87	46.11	46.00	46.00	44.81	44.7	44.7
ADAM	BatchNorm	49.01	35.68	35.68	53.76	52.24	52.24	50.56	47.12	47.12
	Dropout	41.00	39.67	39.67	51.33	48.92	48.92	56.02	52.15	52.15
	No Regularization	41.44	38.33	38.33	50.18	48.23	48.23	48.90	47.35	47.35

Table 1: Tabular description of all the values across all the 18 variants. values are percentages rounded to 2 decimals

3 VGGNet 16:

3.1 Architecture Used:

I have referred the below image to build my VGGNet 16 Network[3].

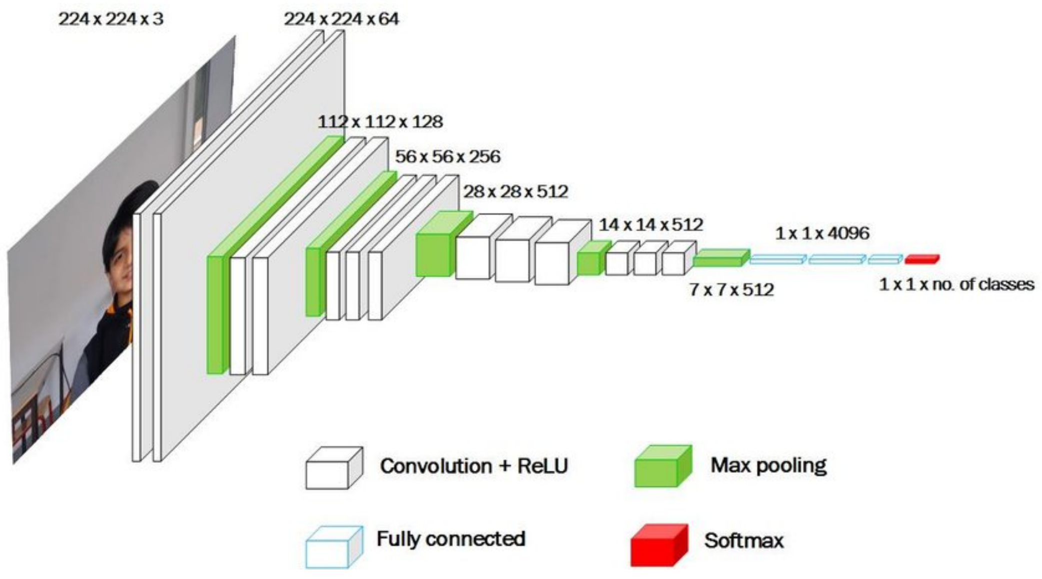


Figure 1: VGGNet16 Architecture

- I have made few changes to the above structure, I have considered to use only half of the filters for each convolution layers i.e. for example instead of 64 I used 32 filters. I did the same for all layers.
- I have changed the strides of max-pooling layers such that I get a decent dimensions for the convolution layers to extract filters/features.
- Using the default architecture with relu resulted in gradient explosion because of the image size in the dataset we use.
- So I am using LeakyReLU with alpha set to 0.1 this setting gave decent results without exploding gradient. I have also used HeNormal to initialize weights across all models.

3.2 VGGNet 16 with SGD optimizer:

- Below are the graphs for the loss and accuracy with SGD as optimizer and with different regularization setting. I have maintained the architecture of the model uniform across all variants so that it is easy to compare the effect of regularizations individually.
- I have used L2 weight regulizer in case of Batch Normalization and Dropout to suppress the loss. In case of No regularization only weights initialization was used.
- In case of No regularization the model over-fitted in just 30 epochs and there is a peak in validation loss which was as expected. The loss stabilized later on because the model does not learn further and thus the loss does not vary much.
- Where as the Dropout helps the model to avoid overfitting on training data, as we can see the accuracy graph of VGG with dropout the accuracy reaches above 90% after 80 epochs. Clearly we can observe the effect of Dropout to avoid the overfitting of data and helps in increasing the accuracy to some extent.
- Here Batch Normalization helps the networks in many ways like avoiding Overfitting, improves the accuracy, helps in suppressing the loss further more.

- we can observe the difference in loss graphs of batch normalization and dropout, the loss in batch normalization is suppressed more than dropout.

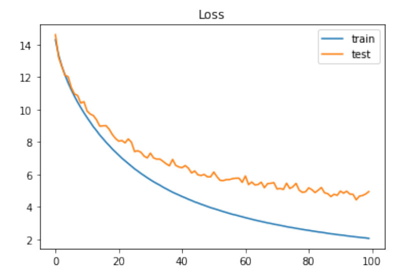
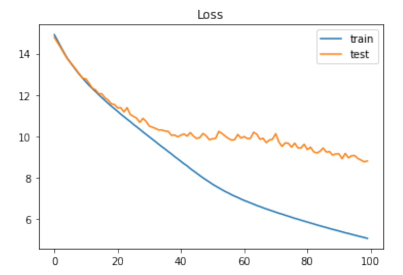
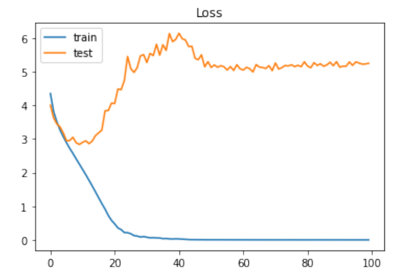
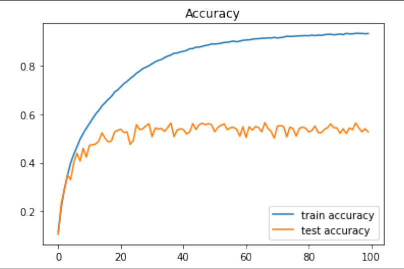
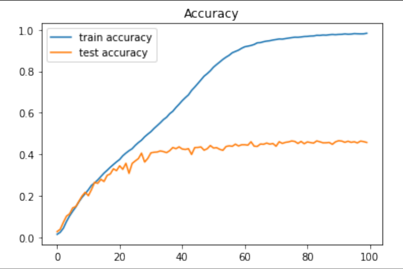
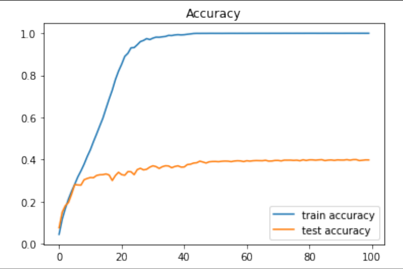
Loss Graph of VGG with BatchNormalization	Loss Graph of VGG with Dropout	Loss Graph of VGG with NO Regularization
		
Accuracy of VGG with BatchNormalization	Accuracy of VGG with Dropout	Accuracy of VGG with NO Regularization
Accuracy : 52.67%	Accuracy : 45.68%	Accuracy : 39.87%
		

Table 2: VGGNet Models with SGD optimizer

3.3 VGGNet 16 with ADAM Optimizer:

- Below are the graphs for loss and accuracy for VGGNet 16 with ADAM optimizer and I have maintained same architecture as mentioned above, for all 3 variants.
- Compared to SGD, ADAM does not overfit the data in case of No Regularization and there are irregularities in the loss but the loss values are less than loss values of SGD and there is no significant difference between the accuracies.
- The sudden peak in loss graph of No regularization is probably because of a bad batch during the training.
- Adam with dropout converged so fast and triggered early stopping before 35 epochs. Here, in this scenario dropout did not perform as expected, instead of avoiding overfitting of data, in contrast it converged fast. I think that's because of the learning rate or probably it might have stuck in a local minima.
- Coming to Batch Normalization graphs there are many irregularities in the loss and even in the accuracy, this is because of the fact that batch normalization tends to increase the learning rate for faster convergence but in case of Adam increasing the learning rate collapses the network, as Adam is supposed to use lower learning rates.
- As I have used the same architecture, I did not decrease learning rate in case of bacth normalization, so that we can understand better how each regularization worked.
- Now we can state that for same setting of the network some regularizations might work and some may not, so we need to individually optimize them again.

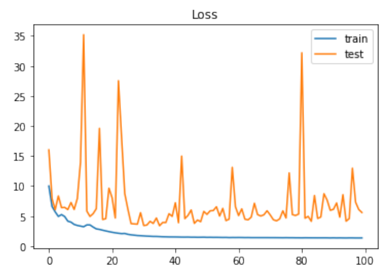
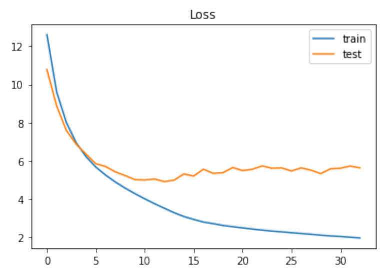
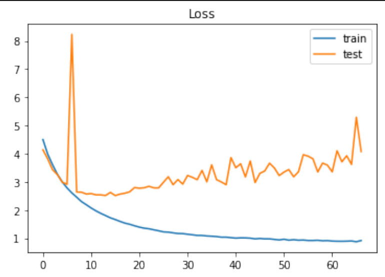
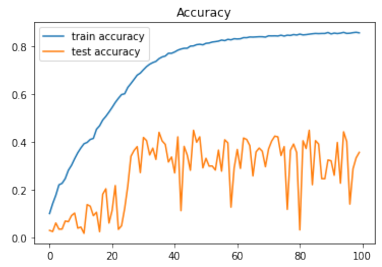
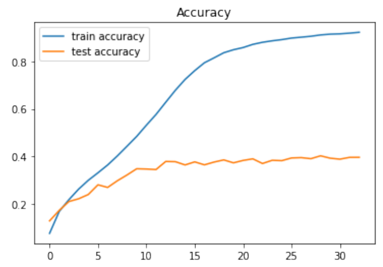
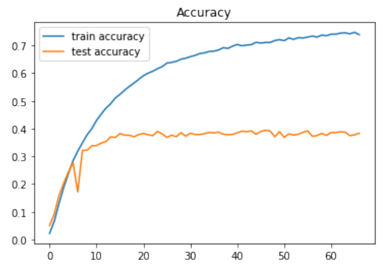
Loss Graph of VGG with BatchNormalization	Loss Graph of VGG with Dropout	Loss Graph of VGG with NO Regularization
		
Accuracy of VGG with BatchNormalization	Accuracy of VGG with Dropout	Accuracy of VGG with NO Regularization
Accuracy : 35.68%	Accuracy : 39.67%	Accuracy : 38.33%
		

Table 3: VGGNet Models with ADAM optimizer

4 ResNet 18:

4.1 Architecture Used:

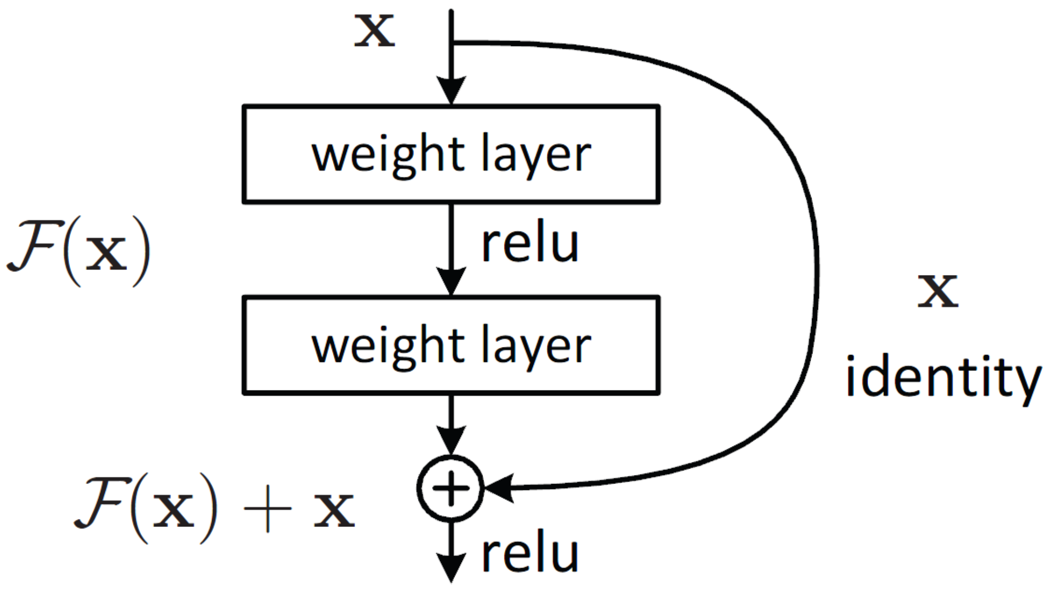

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2: Residual Block.[4]

- Resnet uses blocks shaped as shown in the image above. These are also called “identity shortcut connection", which can skip one or more layers. In our case I have considered to skip only one layer at a time.
- In my implementation I have used 6 such residual blocks with LeakyReLU as my activation function for all the layers. I have placed increased filters across residual blocks as we move further in the network starting at 64 to 512 filters before average pooling.
- I have used max-pooling such that my image dimensions are 8x8 before average pooling.
- I have maintained consistancy in model architecture across all the variants of SGD and ADAM, so that we can understand the comparison.

4.2 ResNet18 with SGD Optimizer

- Below are the Loss and accuracy graphs of ResNet with SGD as optimizer.
- Loss in No regularization increases because no restriction is applied on weights. Training accuracy stabilizes after 40 epochs i.e. overfitting.
- The graphs below are the plots without using weight regularization in any of the models. That is why the loss graphs are irregular without suppression.
- Dropout, did not help the model as expected because the graphs of dropout and no regularization does not vary much in terms of accuracy, even in terms of Loss we can see a little variation on how the loss decreases once the model saturates.
- Batch Normalization, activates early stopping at nearly 70 epochs. It helps the network to improve its performance in terms of accuracy, reaches good accuracy comapred to other two variants.

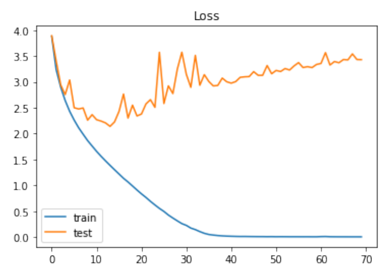
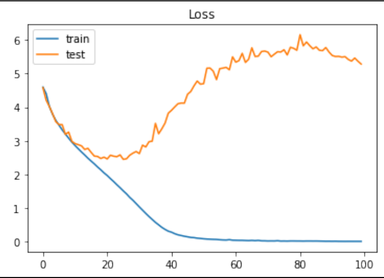
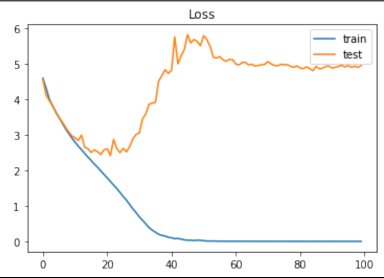
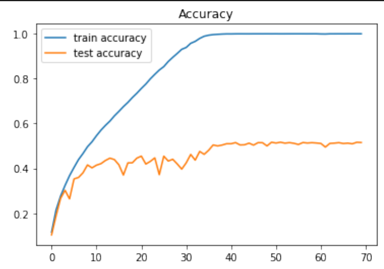
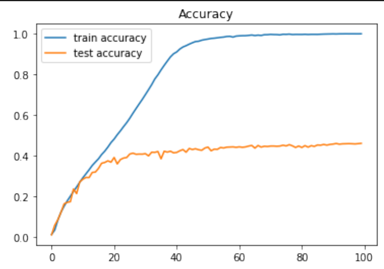
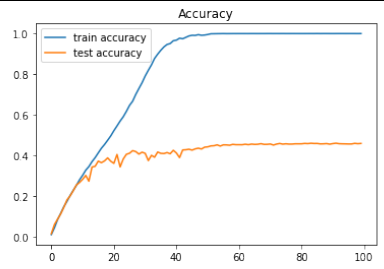
Loss Graph of ResNet18 with Batch-Normalization	Loss Graph of ResNet18 with Dropout	Loss Graph of ResNet18 with NO Regularization
		
Accuracy of ResNet18 with Batch-Normalization	Accuracy of ResNet18 with Dropout	Accuracy of ResNet18 with NO Regularization
Accuracy : 51.68%	Accuracy : 45.97%	Accuracy : 46.00%
		

Table 4: ResNet Models with SGD optimizer

4.3 ResNet18 with ADAM Optimizer

- Below are the graphs for Loss and Accuracy of ResNet with Adam as optimizer. The models does not have L2 regularization.
- In No regularization, the model saturates in around 10-15 epochs, much faster compared to SGD. But does not have smooth curve as SGD.
- In case of Dropout, it improved the validation accuracy when compared to SGD.
- Batch Normalization, helped in faster learning and improved accuracy compared to SGD.
- When 3 variants are considered, batch normalization performed better.

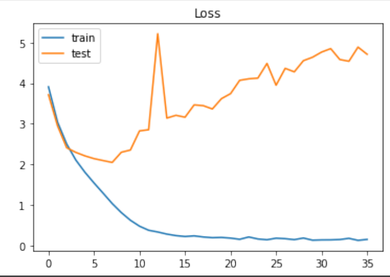
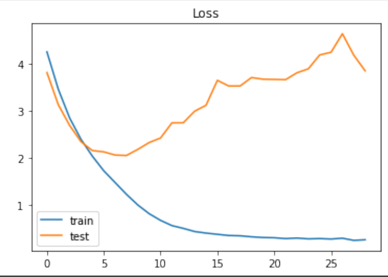
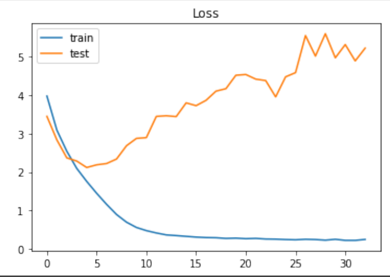
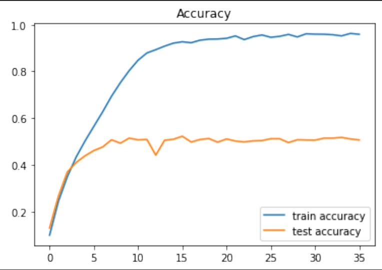
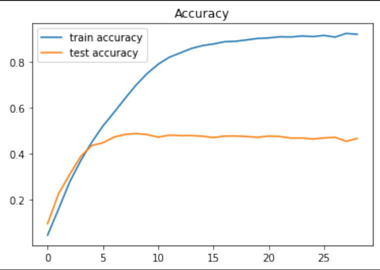
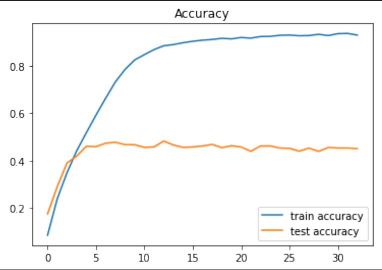
Loss Graph of ResNet18 with Batch-Normalization	Loss Graph of ResNet18 with Dropout	Loss Graph of ResNet18 with NO Regularization
		
Accuracy of ResNet18 with Batch-Normalization	Accuracy of ResNet18 with Dropout	Accuracy of ResNet18 with NO Regularization
Accuracy : 52.24%	Accuracy : 48.92%	Accuracy : 48.23%
		

Table 5: ResNet Models with ADAM optimizer

4.4 Additional experiments done on ResNet:

I have performed a comparison experiment on ResNet. I compared the models with Batch Normalization with L2 regularization and without L2 regularization.

4.4.1 SGD Optimizer

- we can see a clear suppression of Loss in L2 regularization when compared to without L2.
- The sudden peak in middle is due to bad batch of training data samples.
- There is no significant change in accuracy graph.

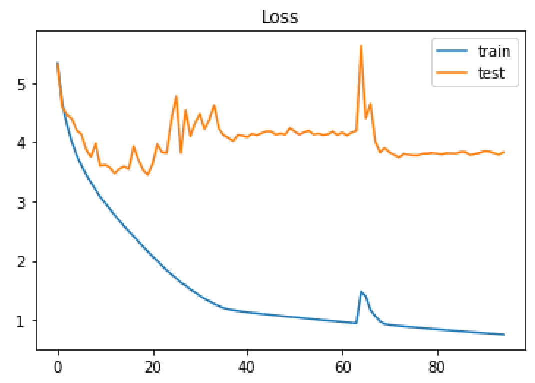
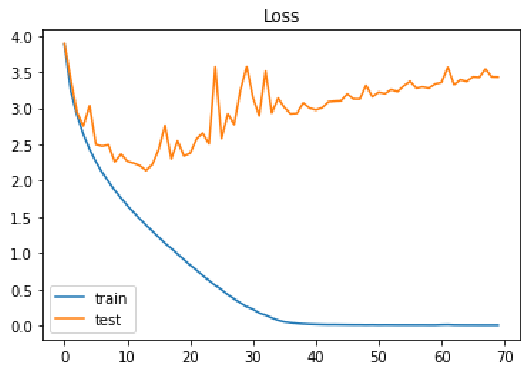
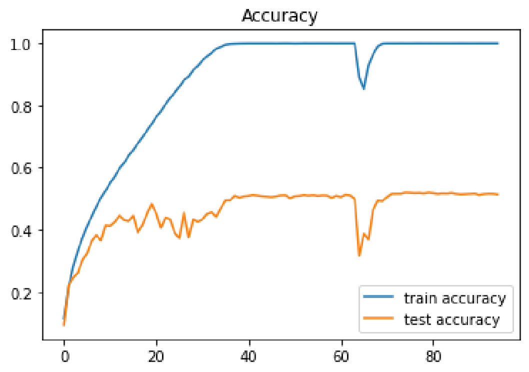
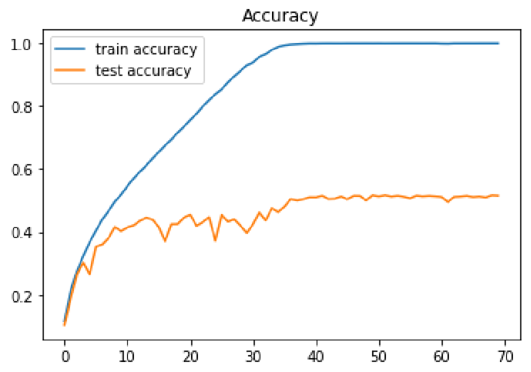
Loss Graph of ResNet18 with BatchNormalization and with L2 regularization SGD	Loss Graph of ResNet18 with BatchNormalization without L2 regularization SGD
	
Accuracy of ResNet18 with BatchNormalization and with L2 regularization SGD	Accuracy of ResNet18 with BatchNormalization and without L2 regularization SGD
Accuracy : 51.37%	Accuracy : 51.68%
	

Table 6: ResNet Models with SGD optimizer

4.4.2 ADAM Optimizer

- We can observe a little suppression on loss in L2 regularization.
- When the accuracy graph is observed, we can see that the model avoid overfitting of the data. L2 regularization provides an added advantage to the network.

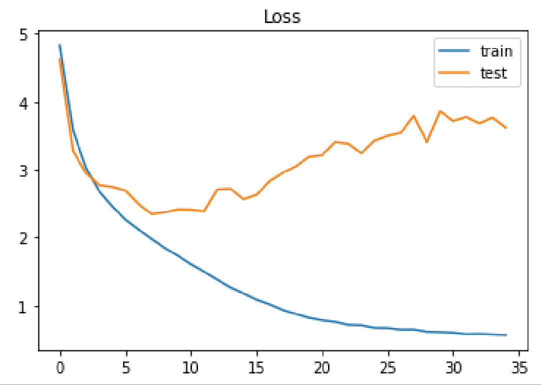
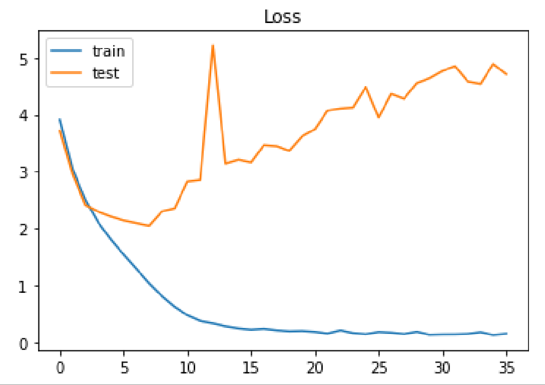
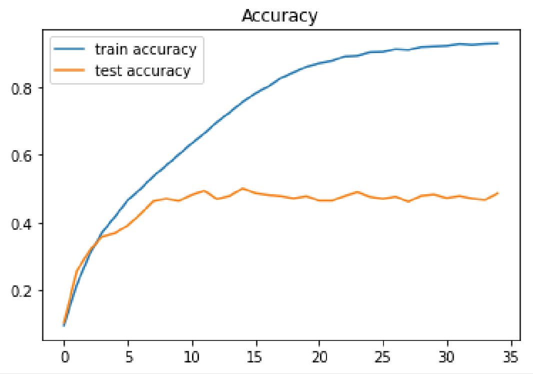
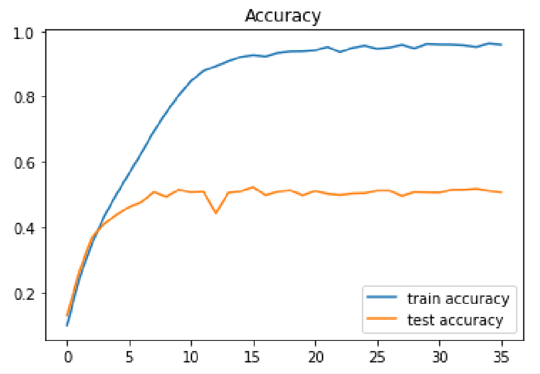
Loss Graph of ResNet18 with BatchNormalization and with L2 regularization ADAM	Loss Graph of ResNet18 with BatchNormalization without L2 regularization ADAM
	
Accuracy of ResNet18 with BatchNormalization and with L2 regularization ADAM	Accuracy of ResNet18 with BatchNormalization and without L2 regularization ADAM
Accuracy : 48.59%	Accuracy : 52.24%
	

Table 7: ResNet Models with ADAM optimizer

5 InceptionV2:

5.1 Architecture Details:

- Below are the three inception blocks used in the implementation of Inception V2. As the image size in the dataset is small I have used only one block per type. For simplicity I am considering 3 blocks as inception_a, inception_b and inception_c as mentioned in the figures below.
- As mentioned in the paper using two 3x3 kernels in series reduces the learning time than 5x5. In addition, according to my intuition the inception blocks allow parallel convolution layers which helps the network to explore more features and aggregate them to train more from a given sample. Thus increases the learning speed and effect of learning.
- I have implemented the inception blocks as mentioned in the Paper[3]. I have changed the filters to 64 in all convolution layers in both inception_a and inception_b. I have used filters of 96 in all convolution layers of inception_c block.
- I have used LeakyReLU as my activation function.
- I have applied max-pooling such that my final image dimensions are 8x8 before connecting to a dense layer before output layer.

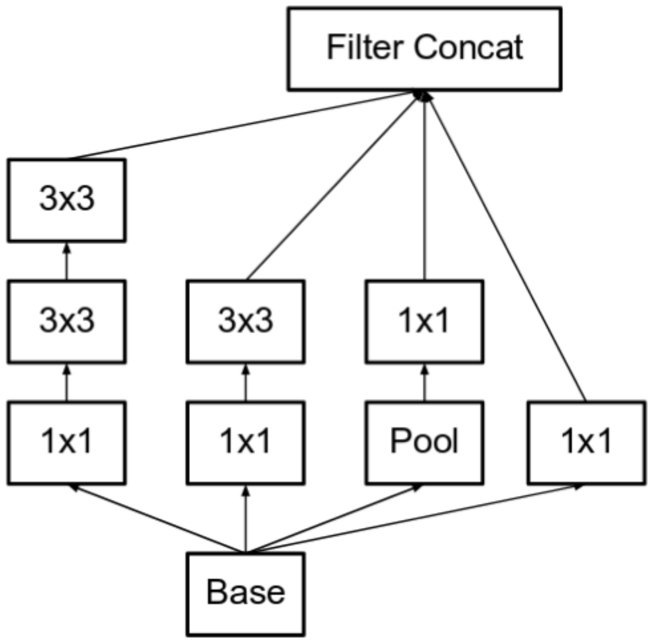


Figure 3: Inception modules where each 5 x 5 convolution is replaced by two 3 x 3 convolution[6]. Inception_a block as mentioned in code

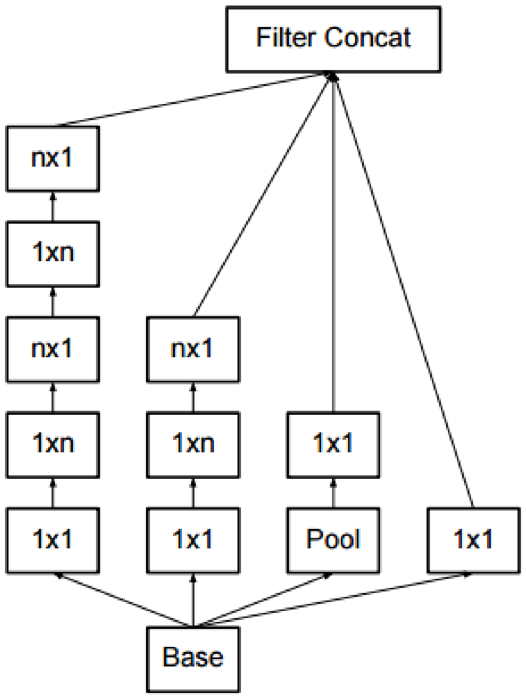


Figure 4: Inception modules after the factorization of the n x n convolutions[6]. Inception_b block as mentioned in code

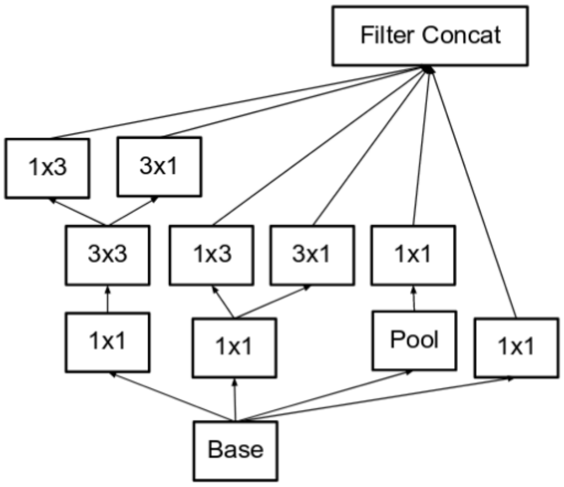


Figure 5: Inception modules with expanded the filter bank outputs[6]. Inception_c block as mentioned in code

5.2 InceptionV2 with SGD Optimizer

- Below are the Loss and accuracy graphs for InceptionV2 3 variants with SGD optimizer. I have also used L2 weights regularization for Batch normalization and Dropout.
- In case of No regularization the validation loss increases because there is no restriction on parameters and the stabilization is due the fact that the model has overfitted and model does not learn. we can observe this in accuracy graph of No regularization there is no change in accuracy after 20 epochs.
- In dropout, the model delays the overfitting to 30 epochs but can not avoid it, this is because the networks are deep enough to capture all the features of the dataset. The loss looks little suppressed because of the L2 regularization.
- In Batch Normalization, it helps the network to improve the training process so it converges in just 10 epochs. And has no improvement in both validation loss and accuracy.

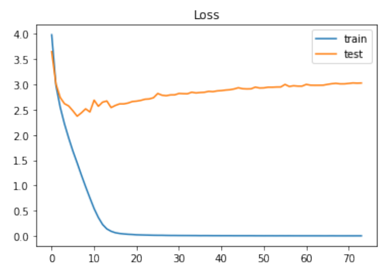
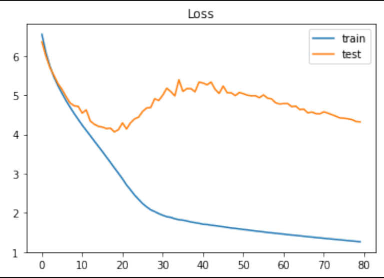
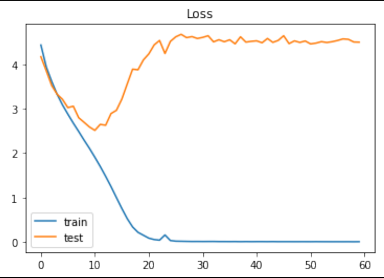
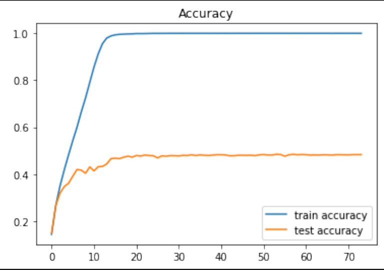
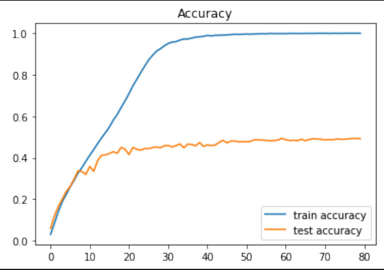
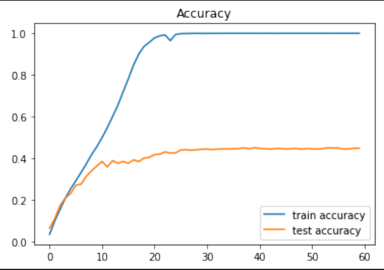
Loss Graph of InceptionV2 with BatchNormalization	Loss Graph of InceptionV2 with Dropout	Loss Graph of InceptionV2 with NO Regularization
		
Accuracy of InceptionV2 with BatchNormalization	Accuracy of InceptionV2 with Dropout	Accuracy of InceptionV2 with NO Regularization
Accuracy : 49.10%	Accuracy : 49.11 %	Accuracy : 44.70 %
		

Table 8: InceptionV2 Models with SGD optimizer

5.3 InceptionV2 with ADAM Optimizer

- Below are the set of graphs of Loss and Accuracy for InceptionV2 for Adam optimizer.
- In case of No Regularization, we can observe the loss graph as expected with increase in loss value and the increase in loss is because of training data overfit.
- In case of No regularization Adam converges much faster than SGD in just 5-10 epochs.
- In Dropout, the model behaves as expected it reduced the overfitting of data and helped in improving the performance of model. The training accuracy increases monotonically describing the accurate training mechanism.
- Batch Normalization does not show much effect on this setting of the inceptionV2. It gives the similar accuracy as the model with No regularization. But when compared to SGD BatchNormalization, in case of Adam it delays the overfitting and makes the training accuracy increase slowly.

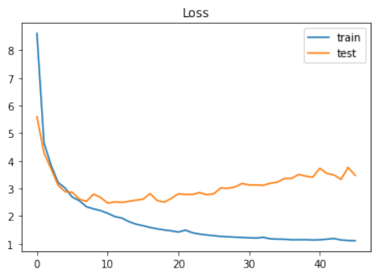
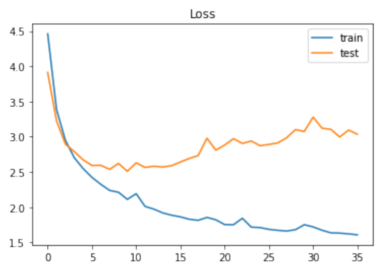
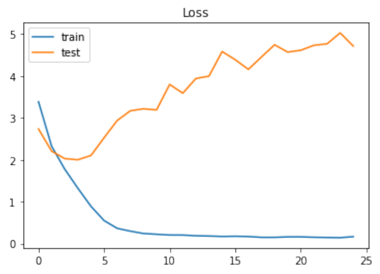
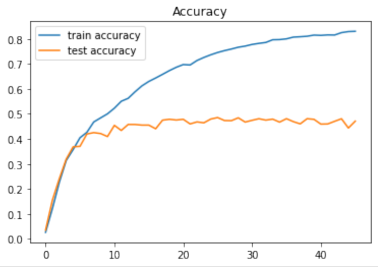
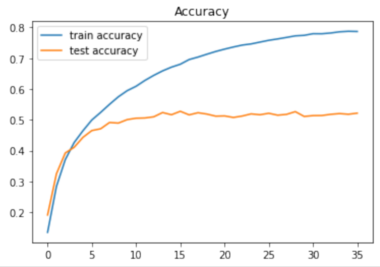
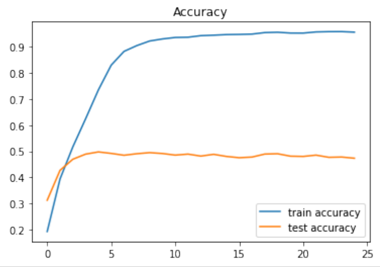
Loss Graph of InceptionV2 with BatchNormalization	Loss Graph of InceptionV2 with Dropout	Loss Graph of InceptionV2 with NO Regularization
		
Accuracy of InceptionV2 with BatchNormalization	Accuracy of InceptionV2 with Dropout	Accuracy of InceptionV2 with NO Regularization
Accuracy : 47.12%	Accuracy : 52.15%	Accuracy : 47.35%
		

Table 9: InceptionV2 Models with ADAM optimizer

6 References

1. https://towardsdatascience.com/batch-normalization-8a2e585775c9
2. https://www.kaggle.com/athota1/vgg16-cifar100-v2?scriptVersionId=12032830
3. VGGNet 16 Architecture
4. https://towardsdatascience.com/building-a-resnet-in-keras-e8f1322a49ba
5. https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624
6. Rethinking the Inception Architecture for Computer Vision
7. https://github.com/sadimanna/neuralnets/blob/master/Inception_v2_on_EMNIST_Digits.ipynb
8. https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c
9. https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5