Assignment-3 (Week-1)

In the first week, we just tried out slight variants of different architectures that we had submitted for the assignment-2(CIFAR-10). We obtained a highest accuracy of **72.03%** on Moodle. We increased the number of Inception modules/layers and the number of iterations as the time limit for this assignment for 6 hours. We tried out different architectures (details of each of them can be found in reports of our assignment-2 submission) and the accuracies obtained are reported below. The details of our best architecture are reported below. The code ran for about 55 mins on Kaggle's GPU.

Architecture	ResNet_Variant	InceptionNet	InceptionNet_2.0 (Best Architecture)	Simple CNN with extra layers
Accuracy	0.6256	0.6912	0.7203	0.6896

A variant of Inception Net with Dropout, Batch Normalization and Data Augmentation

In this architecture, I made a variant of Inception Net with Dropout, Batch Normalization and Data Augmentation using the Model class of Keras. I used Kaggle's GPU for running all of my architectures and all of the accuracies reported here are the ones obtained on Moodle. I obtained a highest accuracy of **72.03%** on the test data using a run time of about 55 minutes on Kaggle. Details of the architecture are below:

1. **Conv2D:** Convolution layer with 64 outputs and Kernel size of 3x3
2. **Conv2D:** Convolution layer with 128 outputs and Kernel size of 3x3
3. **MaxPool:** 2x2 Max pooling layer with Stride =2
4. **Dropout:** Dropout Regularization with dropout rate of 0.1
5. **Conv2D:** Convolution layer with 128 outputs and Kernel size of 3x3
6. **Conv2D:** Convolution layer with 256 outputs and Kernel size of 3x3
7. **MaxPool:** 2x2 Max pooling layer with Stride =1
8. **Dropout:** Dropout Regularization with dropout rate of 0.1
9. **Conv2D:** Convolution layer with 256 outputs and Kernel size of 3x3
10. **Conv2D:** Convolution layer with 512 outputs and Kernel size of 3x3
11. **MaxPool:** 2x2 Max pooling layer with Stride =1
12. **Dropout:** Dropout Regularization with dropout rate of 0.2
13. **Inception Module:** Inception Module with (64,128,64)

14. **MaxPool:** 2x2 Max pooling layer with Stride =2
15. **Dropout:** Dropout Regularization with dropout rate of 0.2
16. **Inception Module:** Inception Module with (64,128,64)
17. **MaxPool:** 2x2 Max pooling layer with Stride =2
18. **Dropout:** Dropout Regularization with dropout rate of 0.2
19. **Inception Module:** Inception Module with (128,256,64)
20. **MaxPool:** 2x2 Max pooling layer with Stride =1
21. **Dropout:** Dropout Regularization with dropout rate of 0.3
22. **FC1:** Fully Connected layer with 512 outputs
23. **Dropout:** Dropout Regularization with dropout rate of 0.3
24. **FC2:** Fully Connected layer with 512 outputs
25. **Dropout:** Dropout Regularization with dropout rate of 0.3
26. **Softmax:** Softmax layer for Classification with 100 outputs

Inception Module: Inception Module consists of performing 3 Conv2D on the previous layer with Kernel size 1x1, 3x3, 5x5 and then concatenating them. (a,b,c) represents number of outputs with 1x1, 3x3, and 5x5 Conv2D respectively.

I also used Batch Normalization after every Convolution layer, Inception Module and after both the Fully Connected Layers. We used eLU(exponential linear unit) as activation function as it gave marginally better results than ReLU. I trained the model alternatively on the augmented data and the real training data. I used Adam Optimizer and Categorical Cross Entropy as the loss function. It ran for an average of 55 minutes and gave the following accuracies on three runs on Kaggle's GPU:

Accuracy	0.7203	0.7192	0.7162

As you can see, it shows a lot of variance, hence we are submitting the one with the 71.92% accuracy.

References:

https://arxiv.org/abs/1409.4842
https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch/
https://arxiv.org/pdf/1511.07289v5.pdf

Nikhil Kapoor - 2017MT10739
Mrigank Raman - 2017MT10736

Assignment-3 (Week-2)

This week me (Mrigank Raman) and my partner (Nikhil Kapoor) had 4 other assignment deadlines. Thus, we did not have enough time to try new architectures. We tried a few things such as introducing another dense layer before softmax, increasing Dropout before these layers and introducing shearing and zooming as data augmentation policies. We were marginally able to improve our accuracy to 73.19% and we are submitting this prediction file.

Mrigank Raman – 2017MT10736
Nikhil Kapoor – 2017MT10739

Assignment-3 (Week-3)

This week me (Mrigank Raman) and my partner (Nikhil Kapoor) tried some image augmentation and enhancement techniques but kept our architecture same. We used unsharp masking on the images and in a way created new data. We concatenated the masked images with the original images to get a dataset containing 1,00,000 images. Unsharp Masking helps to sharpen the edges of an image and thus using this we got increased our accuracy from 73.19% to

74.50%. Then we tried to implement mixup image generator which further increased the accuracy from 74.50% to 74.80%. Then we learnt about cut mix image generator and implemented it which increased our accuracy from 74.80% to 75.38%. We ran our code 4 times with 75.36%, 75.38%, 75.14% and

75.23% being the accuracies. We tried to implement densenet but we did not get enough time to tune the hyperparameters to our liking. We are submitting the prediction file with 75.38% accuracy.

References:
https://github.com/DevBruce/CutMixImageDataGenerator_For_Keras
https://arxiv.org/pdf/1905.04899.pdf
https://arxiv.org/pdf/1710.09412.pdf
https://www.researchgate.net/publication/5597643_Image_Enhancement_via_Adaptive_Unsharp_Masking
https://github.com/yu4u/mixup-generator

Mrigank Raman – 2017MT10736
Nikhil Kapoor – 2017MT10739

Assignment-3 (Week-4)

This week we tried an innovative way of training our CNN. We first trained a classifier to classify the data according to the coarse labels and then built 20 different classifiers to classify it into 5 fine classes of each coarse class. This did not improve our accuracy as we got just around 81% accuracy on coarse labels and it further decreased as we tried to classify it into fine labels. Other than this we did not have much time to new things due to the Diwali break and other assignment deadlines. As such we're happy with our previous submission and are again submitting last week's code and prediction file with 75.38% accuracy.

Mrigank Raman – 2017MT10736
Nikhil Kapoor – 2017MT10739

Assignment-3 (Week-5)

This week, we tried the use of SVMs in our model. Instead of using last dense layer i.e. fully connected layers, we used inbuilt SVM classifier from the sklearn library as the last layer. We used One vs One classifier and trained 100*99/2 classifiers. This did not increase our accuracy, rather decreased our accuracy from 0.7538 to 0.7491 and also due to training 100*99/2 classifiers took longer time as well. We also tried Random Forests as last layer but that too decreased the accuracy to 0.7510. Thus, despite trying plenty of things, we couldn't increase our classification accuracy and hence are submitting the same code as previous week's submission with 0.7538 accuracy.

Mrigank Raman – 2017MT10736
Nikhil Kapoor – 2017MT10739

Assignment-3 (Week-6)

This week, being the last week before majors me and my partner were busy with demos and assignment deadlines. Moreover, we are happy with our previous weeks submission of 75.38% and thus we are submitting the files. Overall we enjoyed trying different thing and we got to learn many different things.

Mrigank Raman – 2017MT10736
Nikhil Kapoor – 2017MT10739