IDS 576 DEEP LEARNING ASSIGNMENT 2

Tarun Kateja (670744115)
Taniya Rajani (658801156)
Shreiya Indulkar (663477095)

Problem 1: Neural Network Design by PyTorch

(a) Training uses mini batch to update the model parameters for each batch of training examples. One Epoch is defined as when full dataset is passed through neural network once. As we need to pass the dataset multiple times to the neural network architecture, we defined number of epochs while running the model. While batches are the subsets of datasets passed through neural network instead of passing whole dataset at once.

(b) Done in train.py

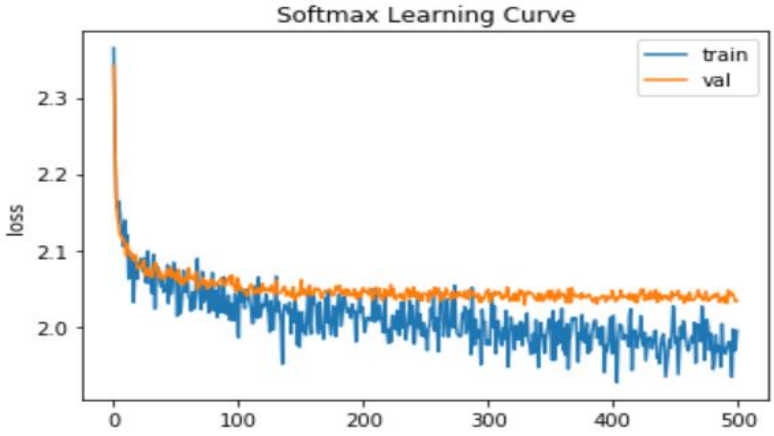
(c) Done in train.py

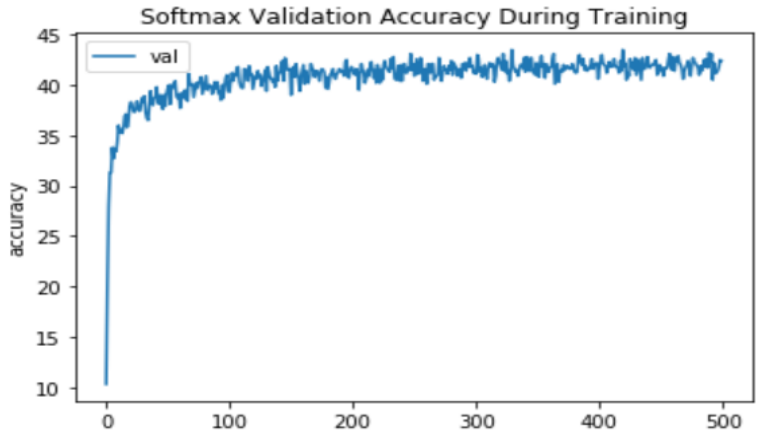
Problem 2: Softmax (Multiclass) Classification

(a) Done in softmax.py

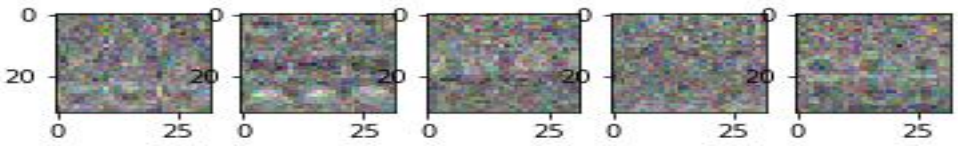
(b) Done in softmax.py

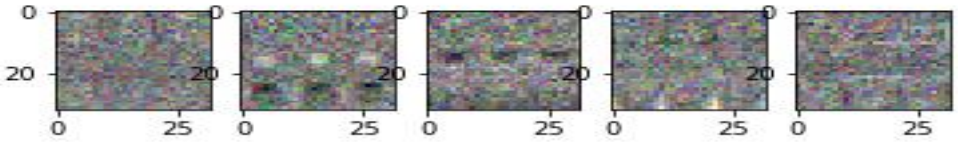
(c) On using log-visualize.ipynb file, below plots of loss vs iterations and accuracy vs iteration are generated using the best model. On varying the hyperparameters, the best parameters for SoftMax are *epochs – 50, weight-decay - 0.0, momentum - 0.0 , batch size - 512 & lr - 0.01.* The best accuracy with these parameters is *40.02%.*

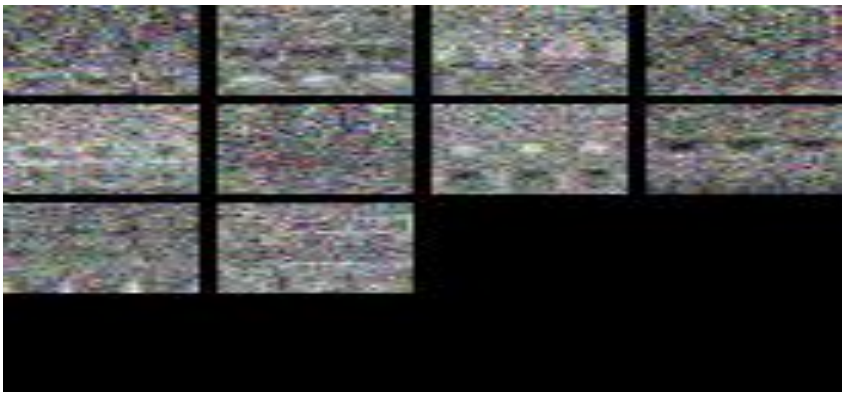




(d) Visualizing the learned weight parameters between input and output layer.



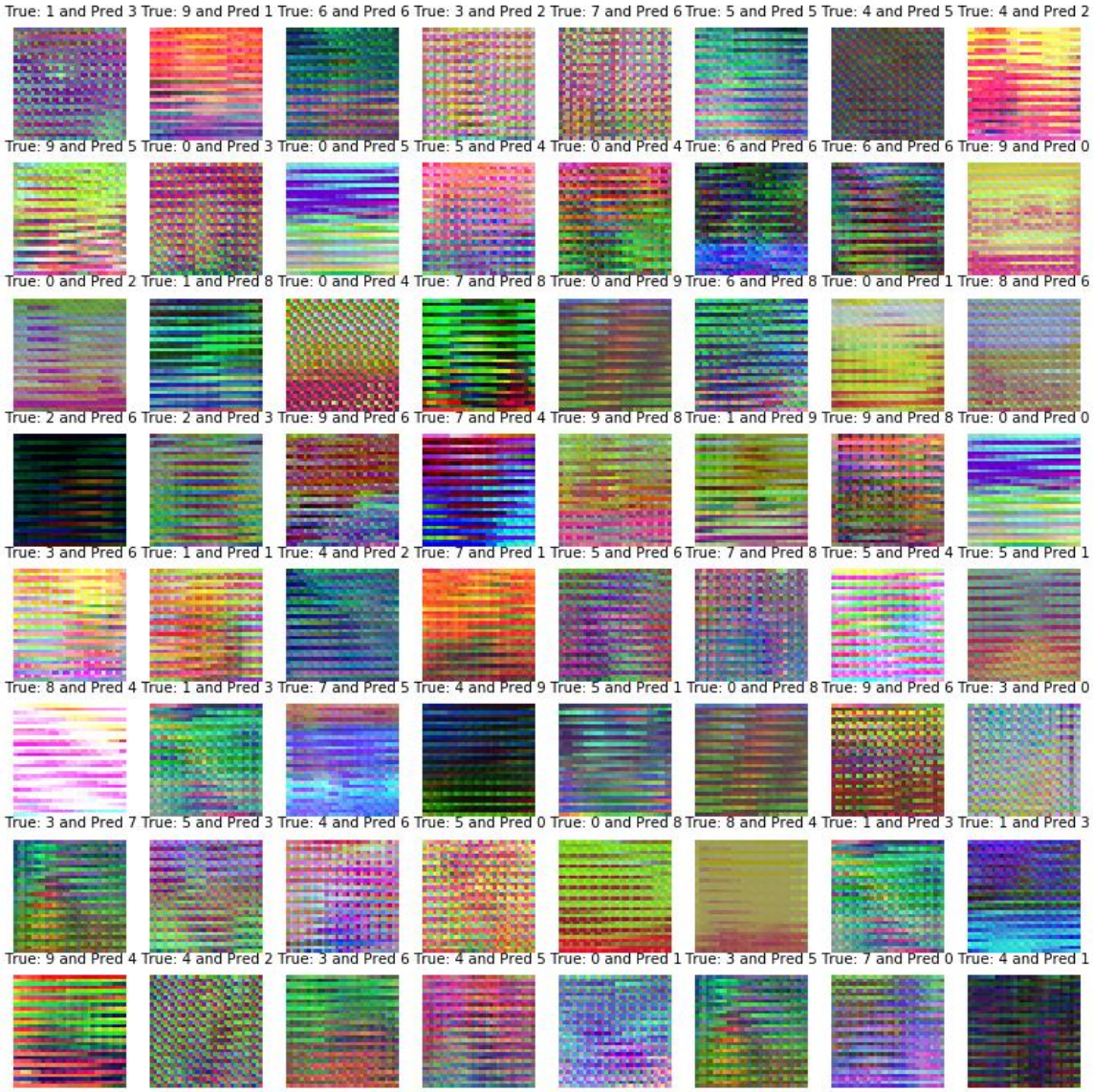




(e) There are around 6000 misclassified images and below are the 64 examples misclassified images. Title of these images signify true and predicted class by the model. CIFAR-10 images are difficult to visualize due to small image size (32*32 colored images). On increasing the aspect ratio, the images are blurred. However, we have tried our best to plot and visualize it. Best performance optimizer is SGD. (Code for this is included in train.py in evaluate function)

There are very interesting and exciting observations in below images. For ex. The first image which has been predicted as class 3 (A bird) but is actually class 1 (An Airplane), now this makes sense in terms of features our model would have learnt. There are more examples where our model got confused between bird, ship and airplane and truck and car and cat with dog or horse.

Wrongly classified image

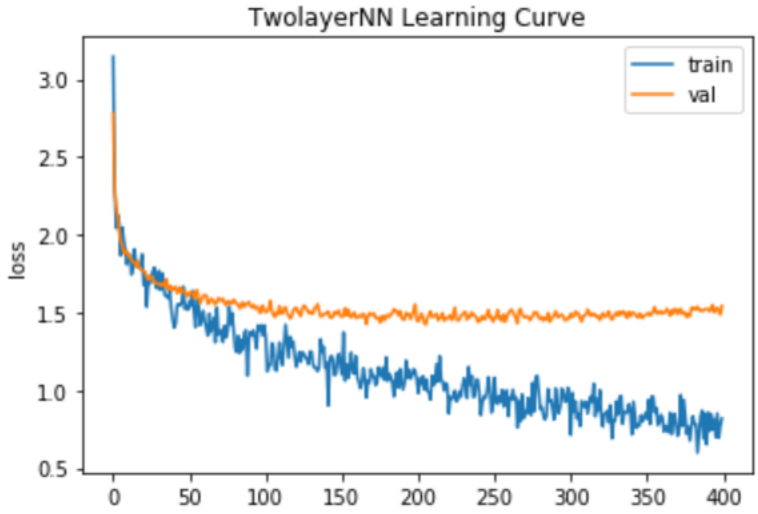


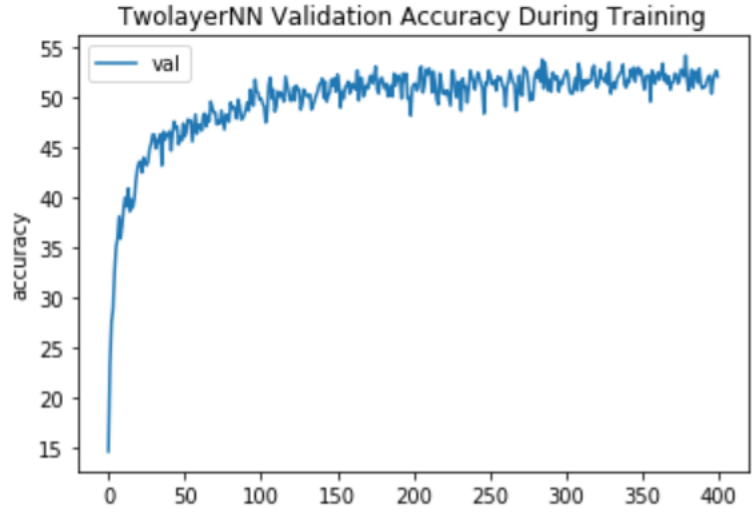
Problem 3: Two-layer Neural Network

(a) Done in twolayernn.py

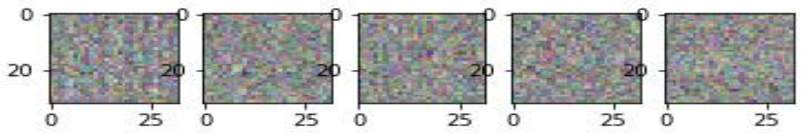
(b) Done in twolayernn.py

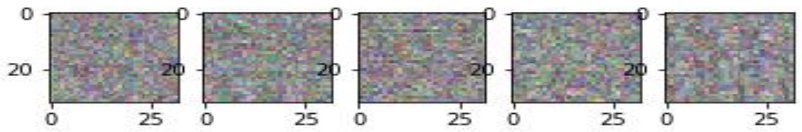
(c) On using log-visualize.ipynb file, below plots of loss vs iterations and accuracy vs iteration are generated using the best twolayerNN model. On varying the hyperparameters, the best parameters are *epochs – 50, hidden dim - 250, weight-decay - 0.0001, batch size - 256 & lr - 0.0001.* The best accuracy with these parameters is *51.02%.* We experimented with multiple hyperparameters and optimizer and got the best results with **Adam** optimizer.

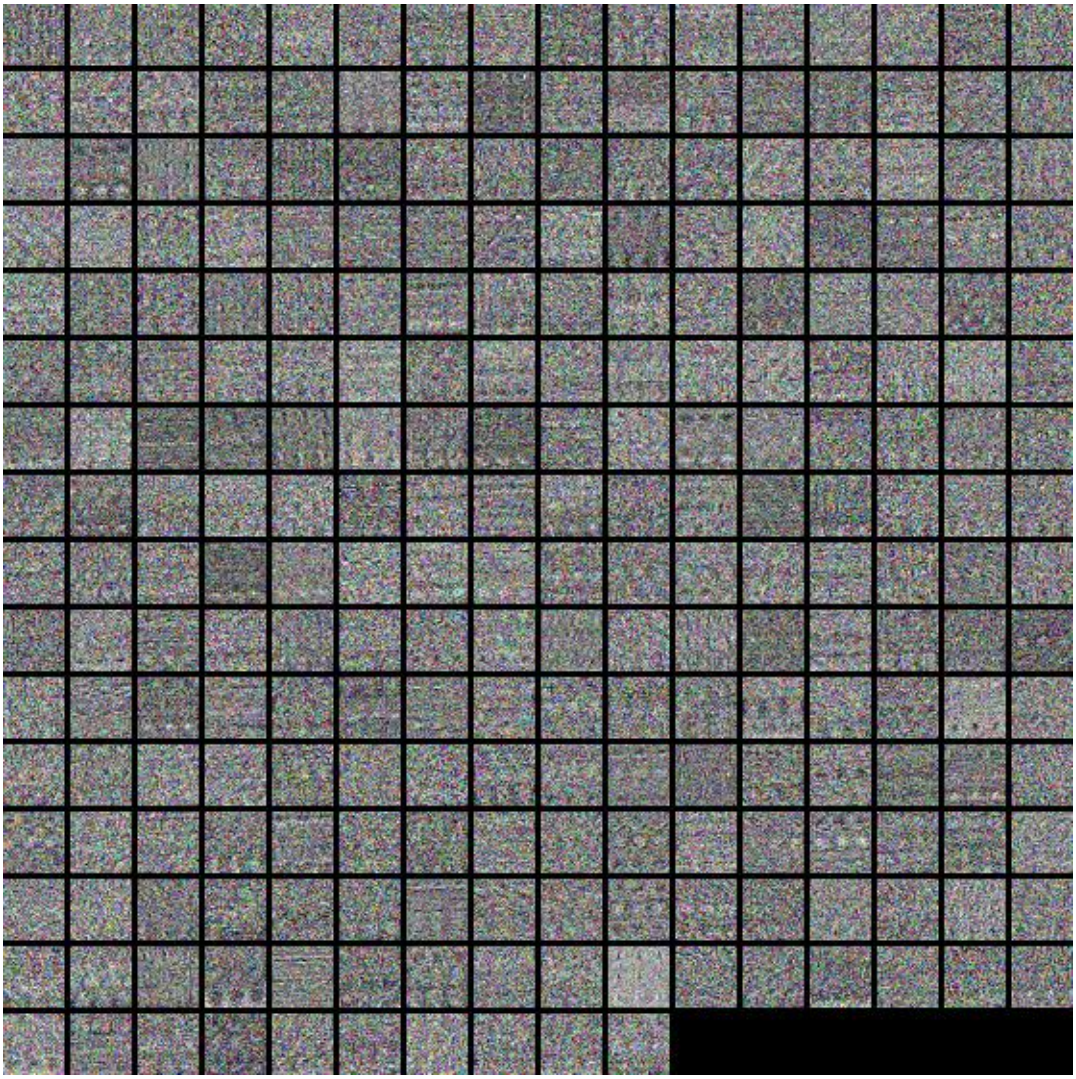




(d) Visualizing the learned weight parameters between input and output layer for twolayerNN.

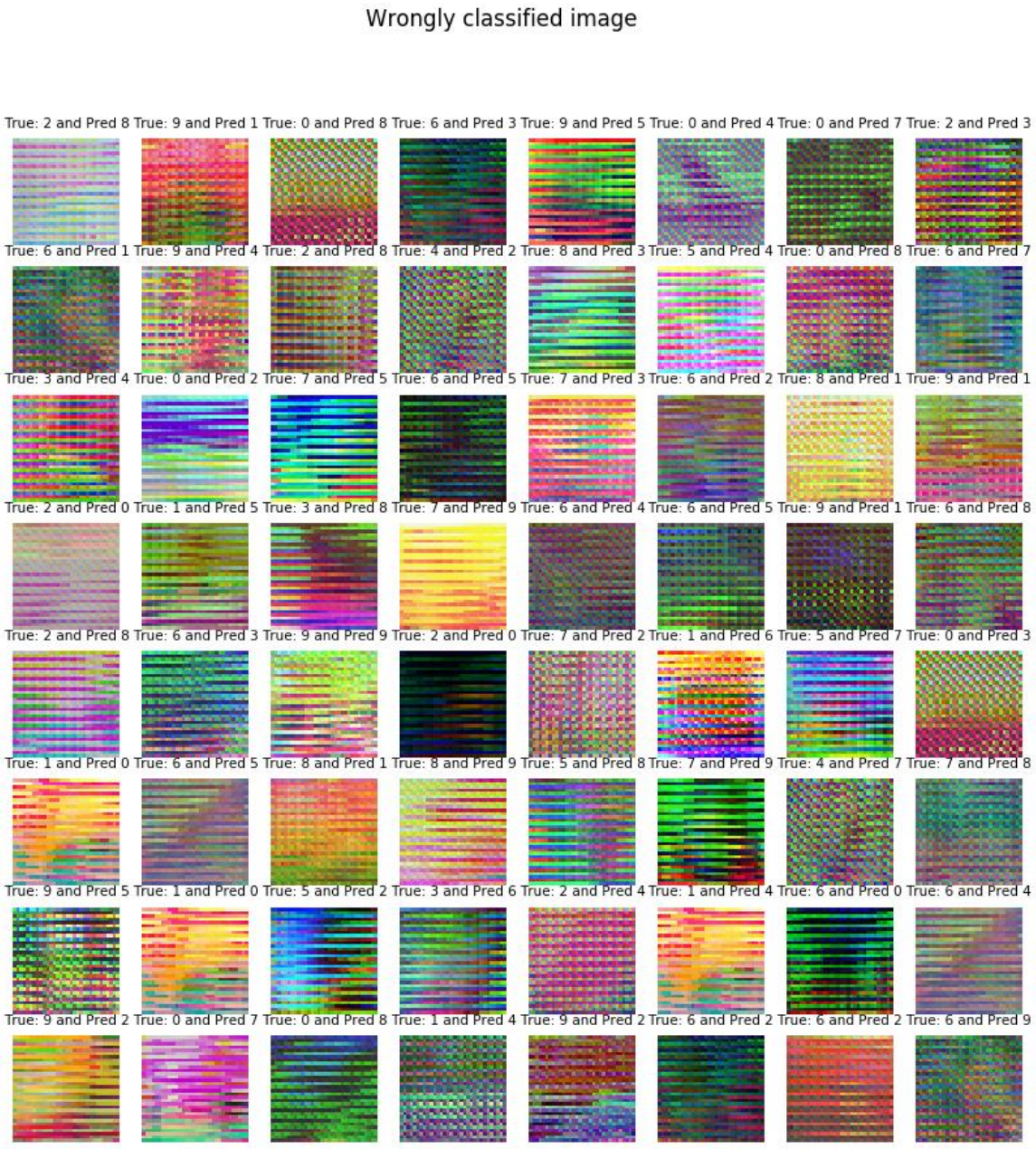






(e). There are around 5000 misclassified images and below are the 64 example misclassified images. Title of these images signify true and predicted class by the model.

The exhaustive answer about approach is provided in answer 2.e. Here also we see similar results where model got confused between bird and ship (image 1). However, as our model has better accuracy, it would have now made lesser mistakes.

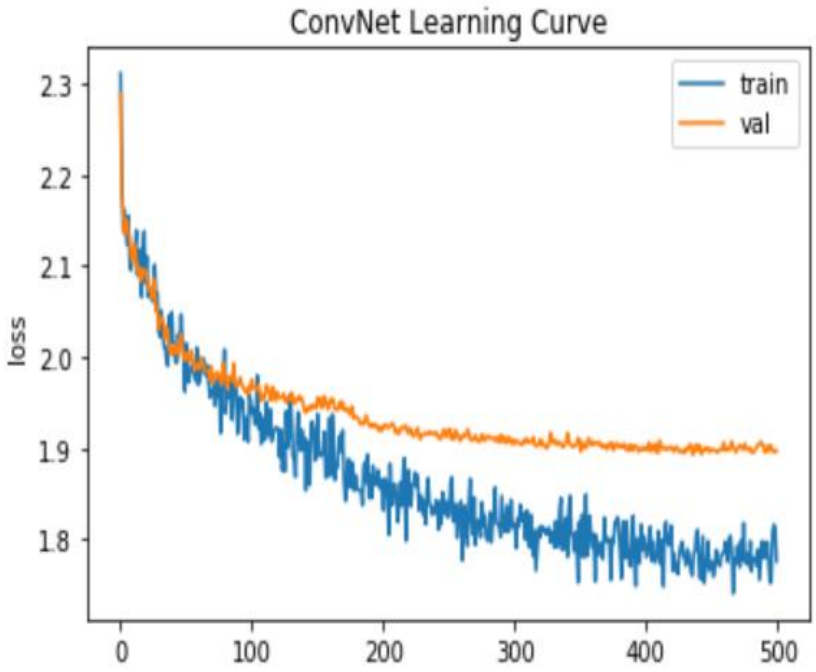


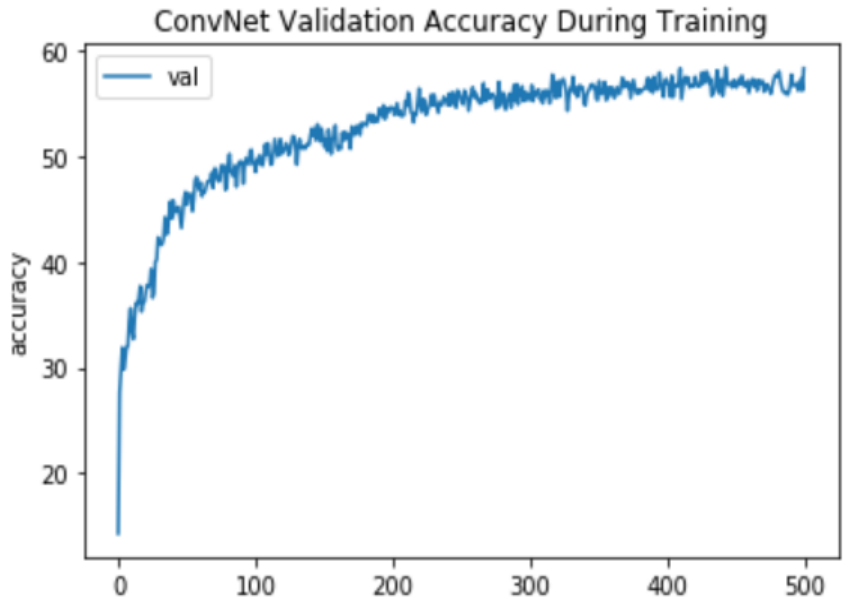
Problem 4: Convolutional Neural Network

(a) Done in convnet.py

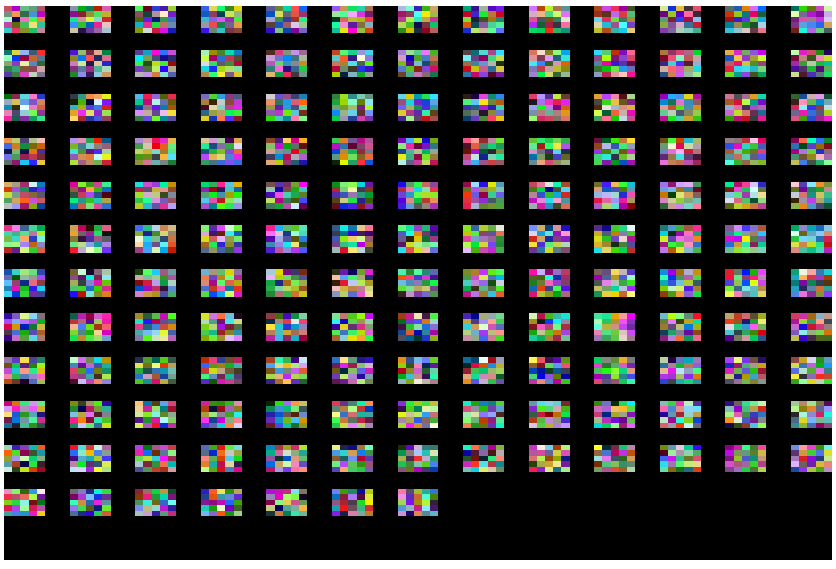
(b) Done in twolayernn.py

(c) On using log-visualize.ipynb file, below plots of loss vs iterations and accuracy vs iteration are generated using the best CNN model. On varying the hyperparameters, the best parameters kernel-size -5, hidden dimension – 50, epochs – 50, weight-decay – 0.01, momentum – 0.9, batch size – 512, lr – 0.001. The best accuracy with these parameters is 58.36%. Best performance optimizer is SGD.

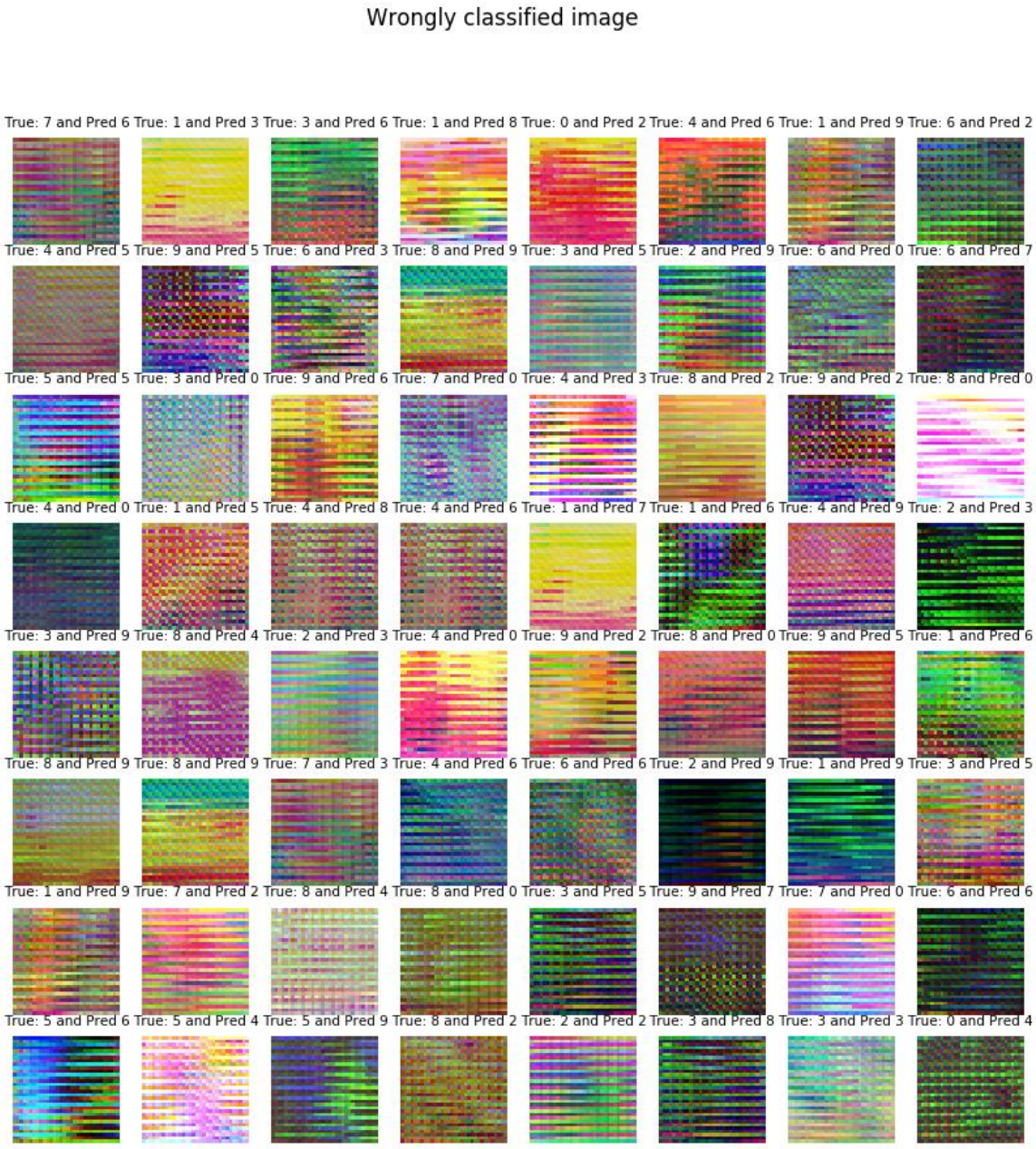




(d) Visualizing the learned weight parameters between input and output layer for convnet. As we can see compared to previous two models, the weight matrix are cleaner in this case as our model has improved significantly



(e) There are around 4200 misclassified images and below are the 64 example misclassified images. Title of these images signify true and predicted class by the model.

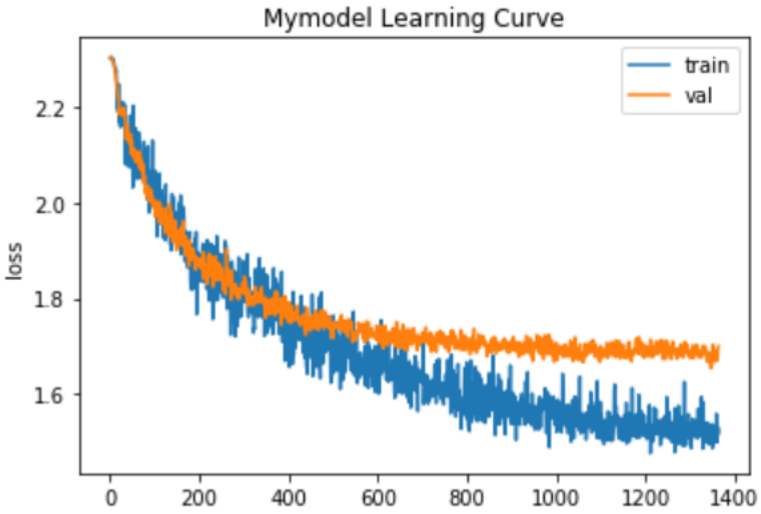


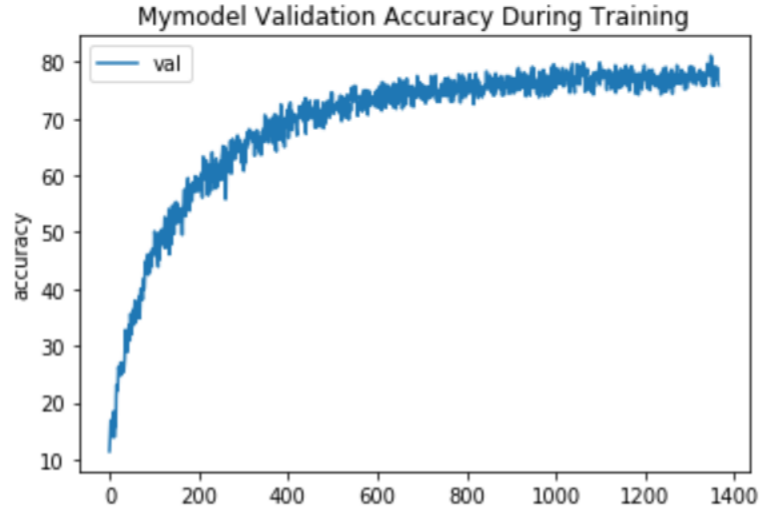
Problem 5: Build Your Own Neural Network

(a) Done in mymodel.py

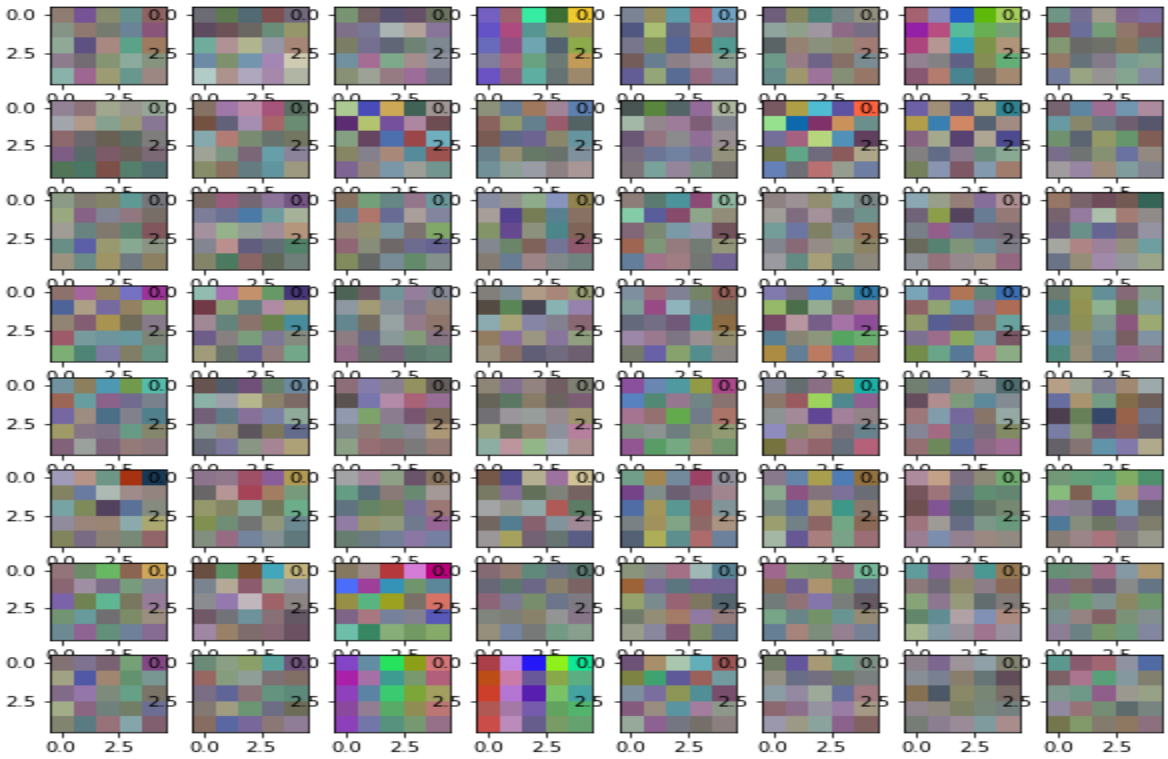
(b) Done in mymodel.py

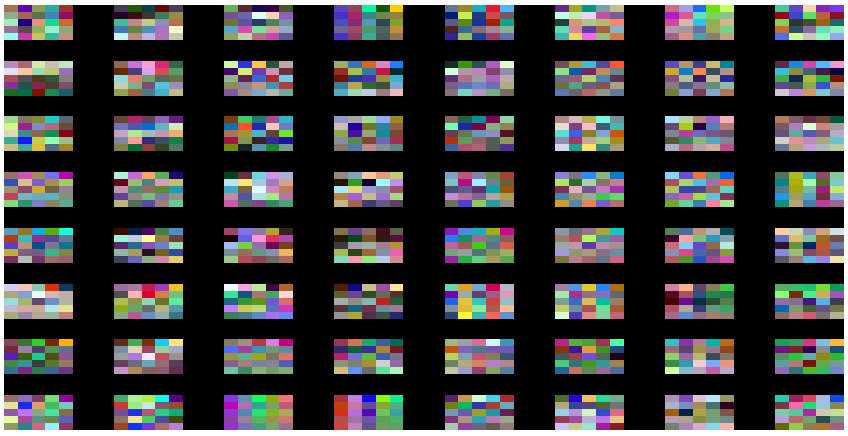
(c) On using log-visualize.ipynb file, below plots of loss vs iterations and accuracy vs iteration are generated using the best mymodel. On varying the hyperparameters, the best parameters are kernel-size 5, hidden-dim 64, epochs 35, weight-decay 0.001, momentum 0.98, batch-size 128 & lr 0.001. The best accuracy with these parameters is 76%.





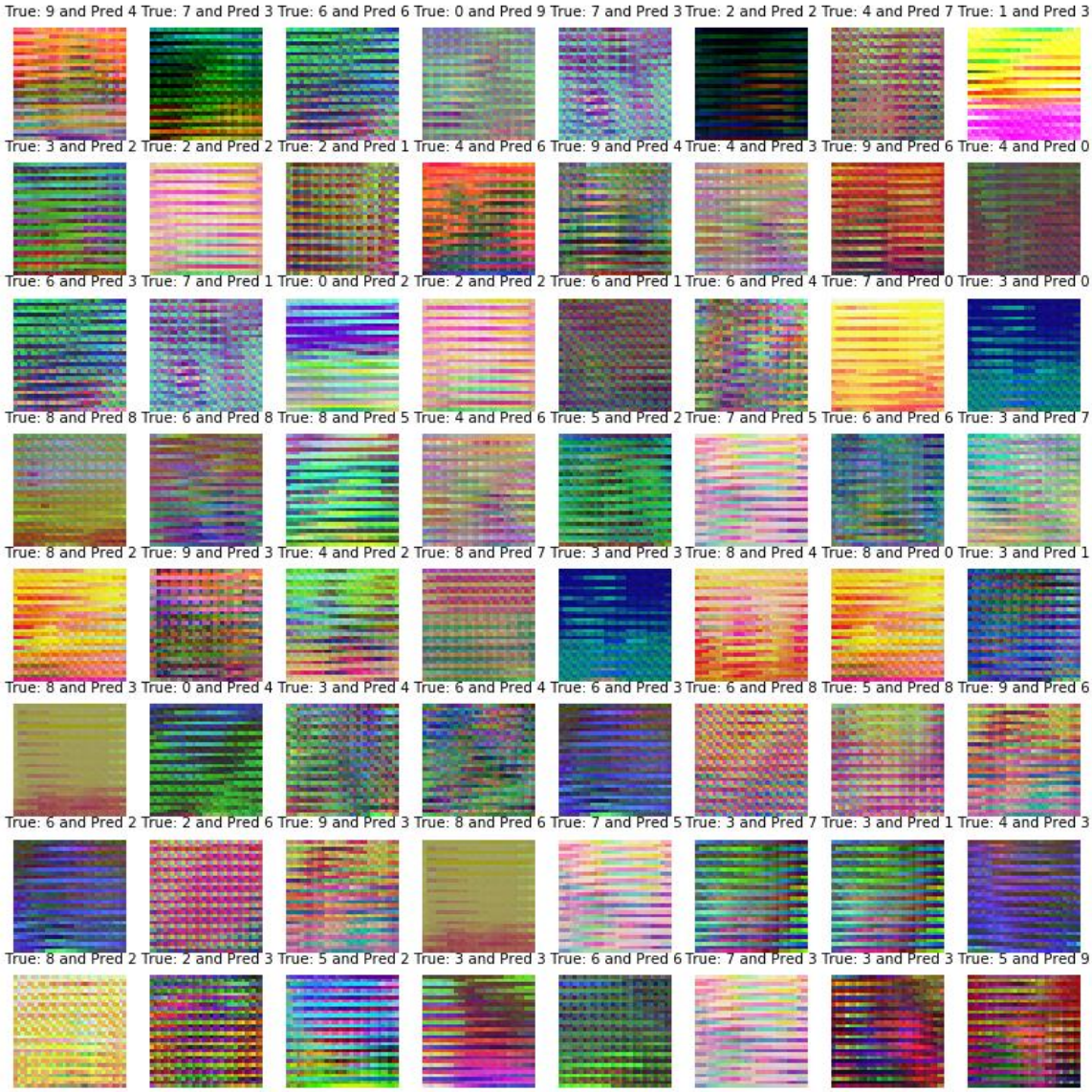
(d) Visualizing the learned weight parameters between input and output layer for mymodel





(e) There are around 2400 misclassified images and below are the 64 example misclassified images. Title of these images signify true and predicted class by the model. A very interesting insight is, model is less confused than previous cases as our accuracy has improved a lot but if we analyze below images, for ex image 1 (Predicted class 4 and actual class 9), the model is confused between a Truck and a Deer and it must be a very bad and blur image (as truck and deer would least likely have common features)!

Wrongly classified image



Problem 6: Transfer Learning

(a) This was a very interesting problem. For our first two models (softmax and 2layer NN), we have simply connected our model to final fc layer of ResNet-18 (We have made sure that we have freezed the parameters of ResNet as question requires us to extract layers from ResNet and use it as initial representation which is nothing but what we have done). For convnet and mymodel, we have created a new model code (resnet_conv.py and resnet_mymodel.py, which needs to be put in model folder like other provided codes for four models). We have created a combine class in these scripts to combine ResNet to our created convnet and mymodel. This helped us to avoid methods like upsampling (we performed 2-sided padding) and we also overcame the dimensionality mismatch issues. Our performance is also very good and as expected for these models with similar and small run time required. Also, as question asks to combine our created model instead of creating new, we satisfy that condition as well!

As CIFAR 10 images are very small compared to ImageNet (for which ResNet was designed), for convnet we have removed final fc, avg pool and layer 5 from ResNet to combine it with our designed convnet and for mymodel (which has 3 convolutional and 3 pooling followed by 2 layer NN), we have removed final fc, avg pool and layer 3, 4, 5 from ResNet to perform the required task.

Please refer to scripts with name (train_resnet_soft.py, train_resnet_2layer.py, train_resconv.py, train_resnet_mymodel.py)

Models for softmax and 2 layer NN is created in training file itself. While separate model files are created for convnet (resnet_conv.py) and mymodel (resnet_mymodel.py) to align to questions. These scripts have 2 classes, one same as previous questions and one class with name combine to combine resnet to our previous built models.

Model Results:

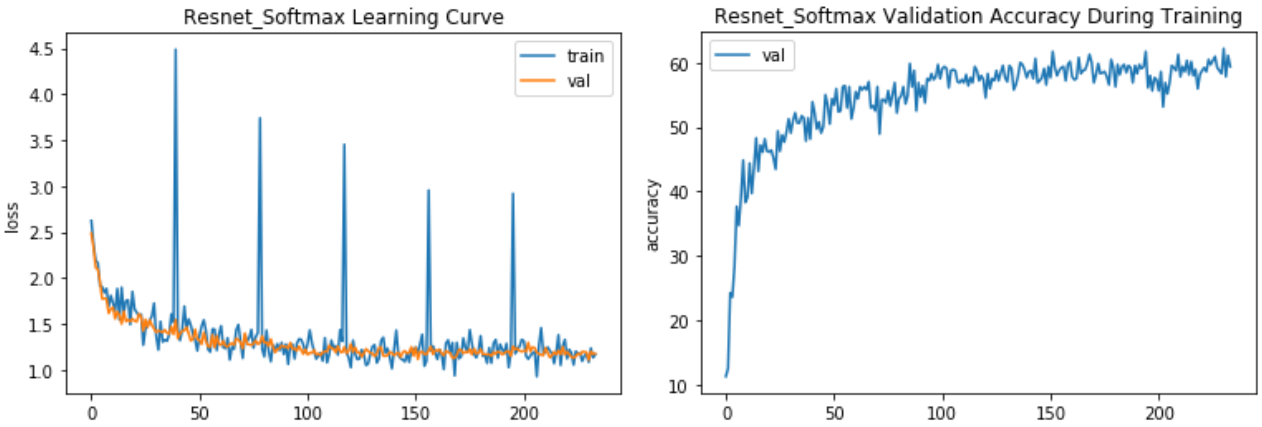
ResNet Models	Accuracy	Previous Models	Accuracy
Resnet + Softmax	58.35%	SoftMax	40.02%
Resnet + 2layer NN	58.58%	2layerNN	51.02%
Resnet + Convnet	68.20%	Convnet	58.9%
Resnet + MyModel	79.17%	Mymodel	76%

Model tuned hyper-parameters:

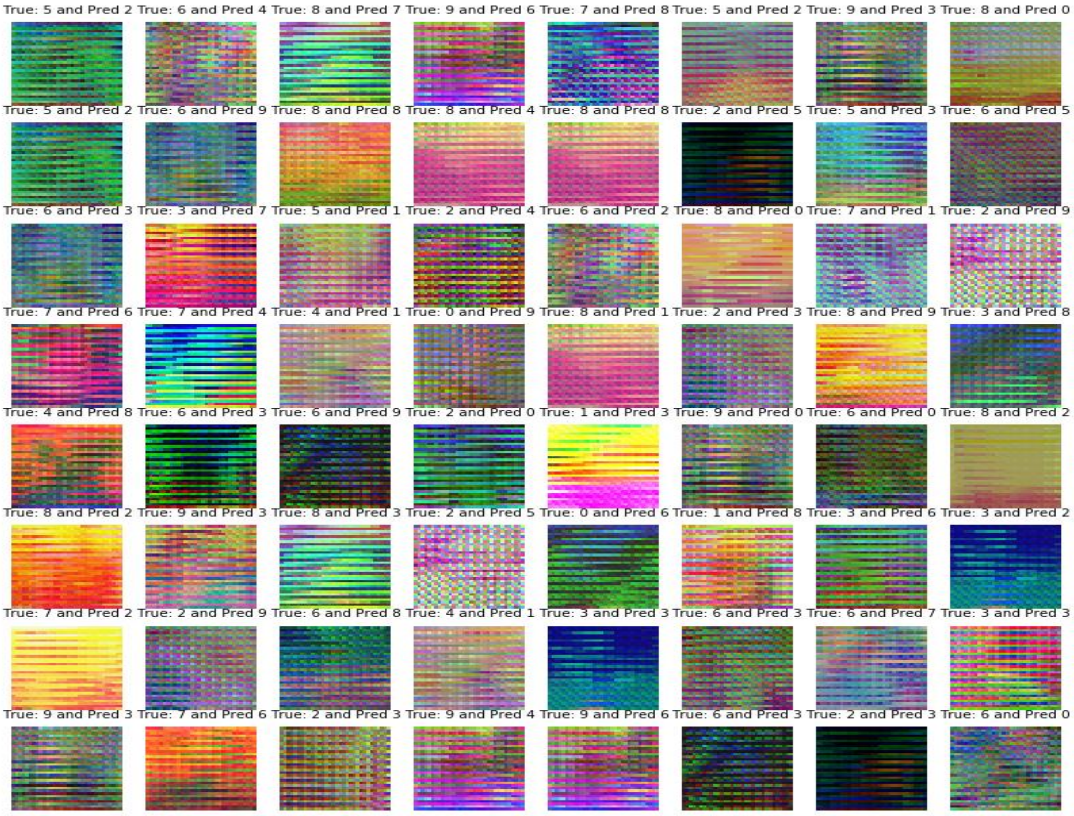
Models	Best Parameters
Resnet + Softmax	--epochs 6 --weight-decay 0.0001 --momentum 0.99 --batch-size 128 --lr 0.001
Resnet + 2layer NN	--hidden-dim 100 --epochs 6 --weight-decay 0.001 --momentum 0.7 --batch-size 128 --lr 0.03
Resnet + Convnet	--kernel-size 3 --hidden-dim 150 --epochs 20

	--weight-decay 0.0005 --momentum 0.9 --batch-size 128 --lr 0.05
Resnet + MyModel	--kernel-size 3 --hidden-dim 128 --epochs 50 --weight-decay 0.0001 --momentum 0.85 --batch-size 128 --lr 0.04

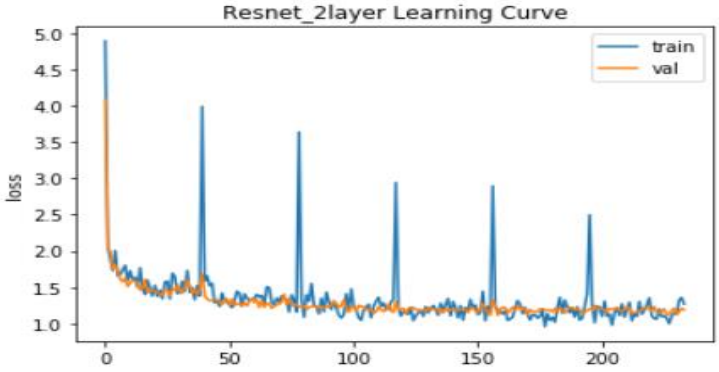
Log visualization plots for ResNet + Softmax

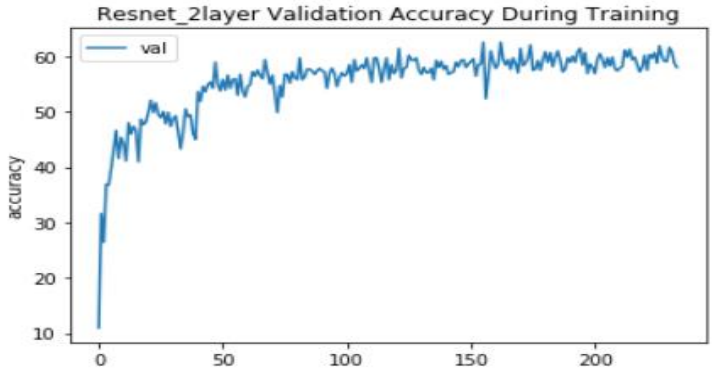


Wrongly classified image

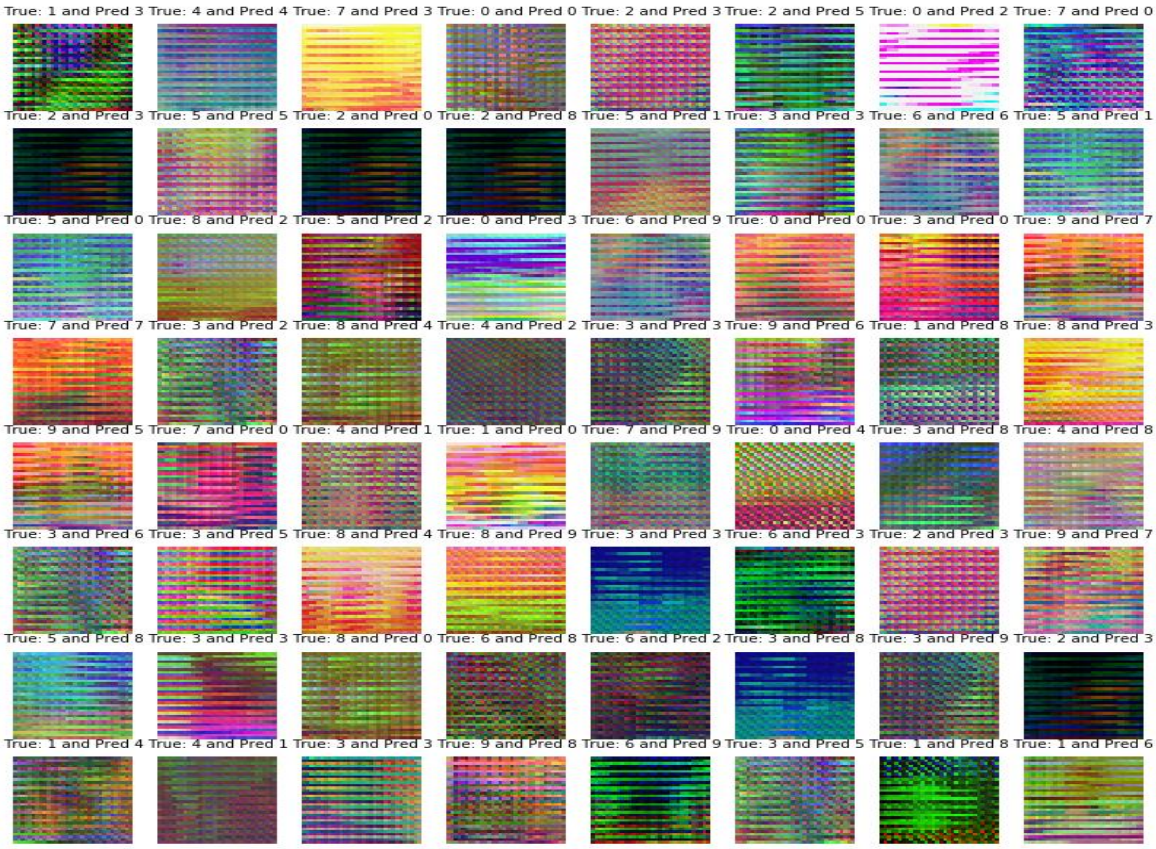


Log visualization plots for ResNet + 2-layer NN

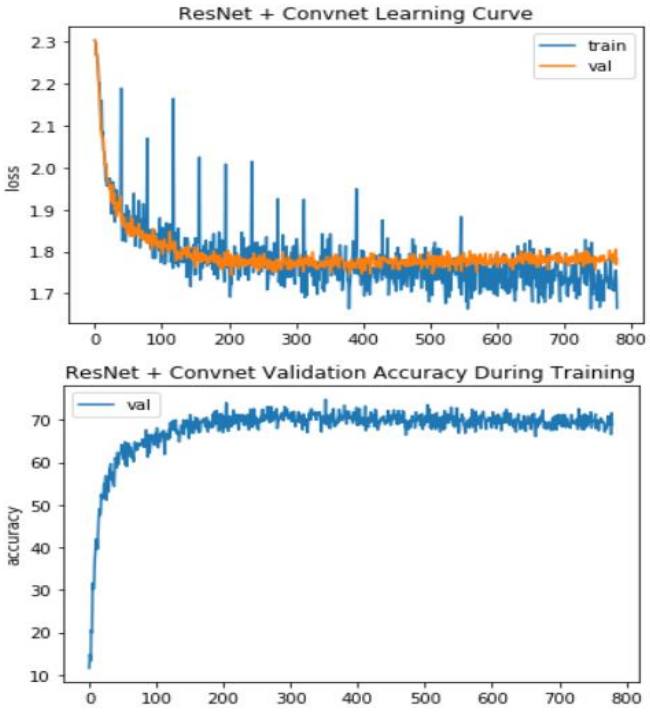




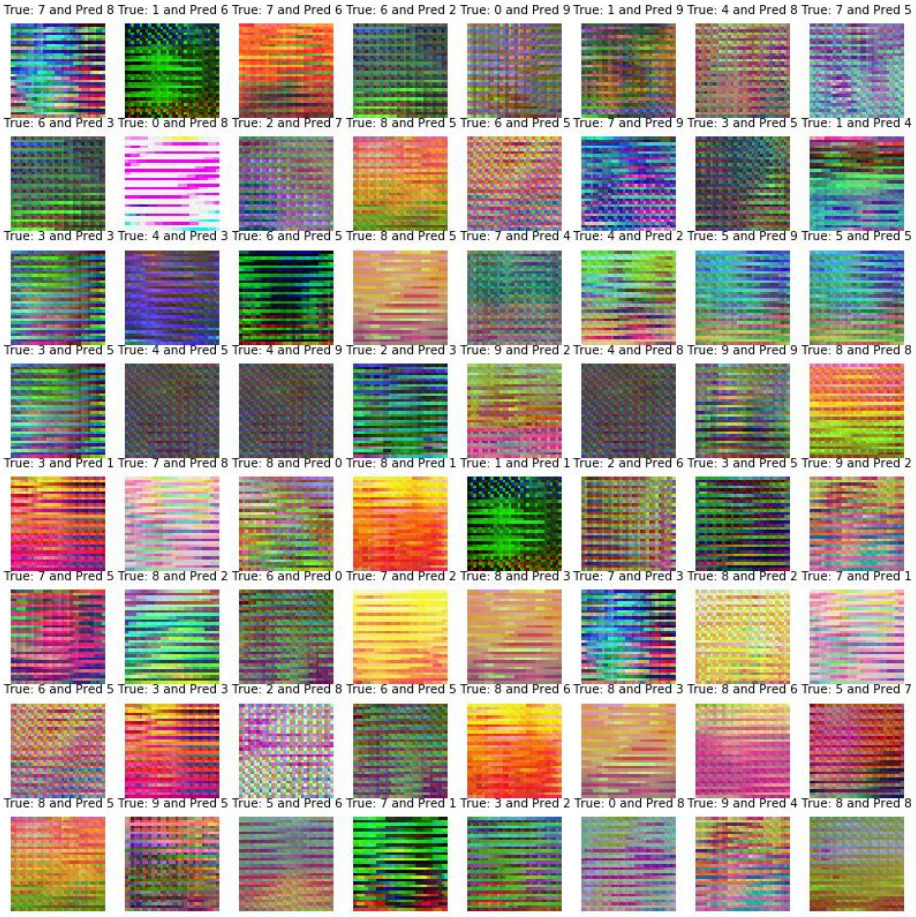
Wrongly classified image



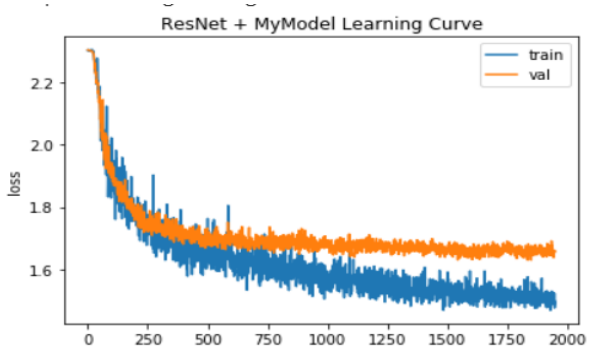
Log Visualization plots for ResNet + Convnet

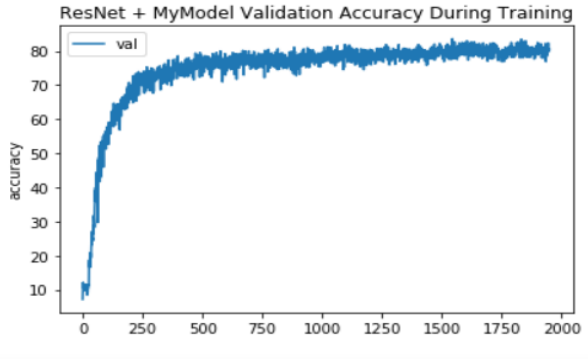


Wrongly classified image

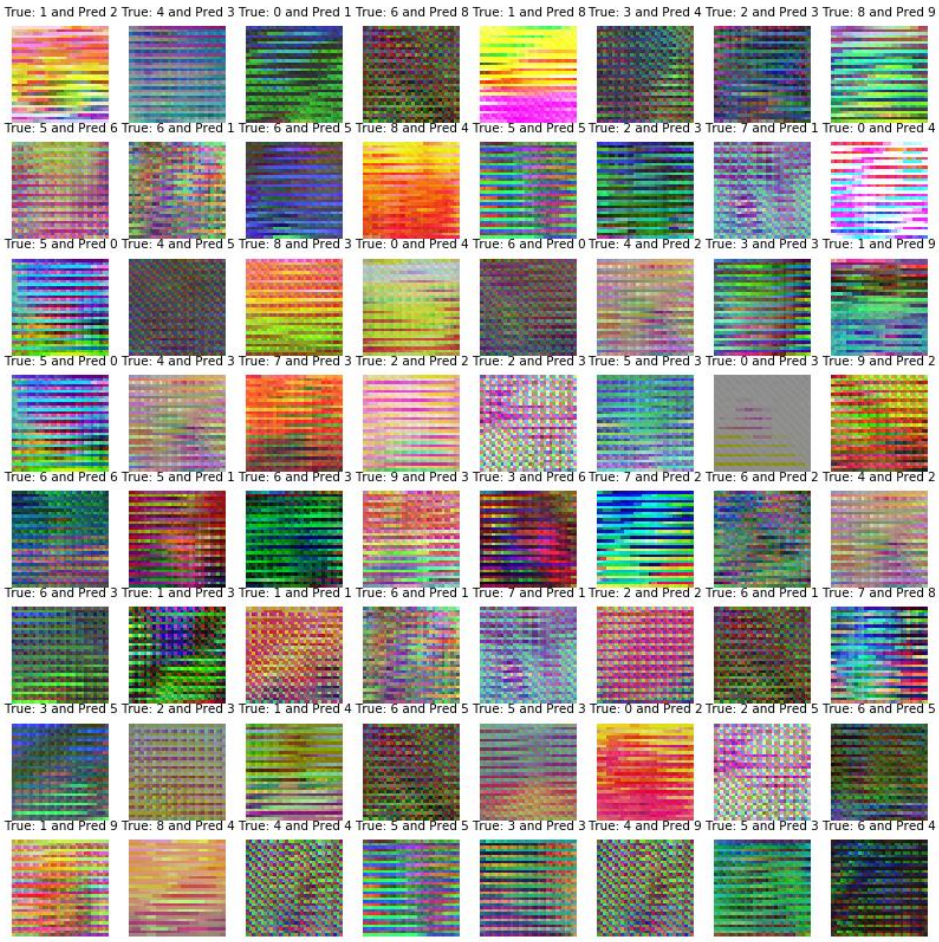


Log visualization plots for ResNet + MyModel





Wrongly classified image



(b) ResNet's initial representation as updatable parameters:

We expected and have drastically improve out performance as we are using ResNet architecture, but parameters are updatable, and model will learn best parameter with really good model architecture for CIFAR 10 dataset.

Please refer to scripts with name (train_resnet_soft_B.py, train_resnet_2layer_B.py, train_resconv_B.py, train_resnet_mymodel_B.py)

Models for softmax and 2 layer NN is created in training file itself. While separate model files are created for convnet (resnet_conv_B.py) and mymodel (resnet_mymodel_B.py) to align to questions. These scripts have 2 classes, one same as previous questions and one class with name combine to combine resnet to our previous built models.

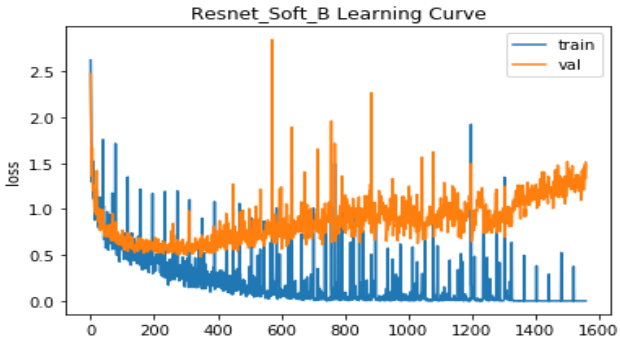
Model Results:

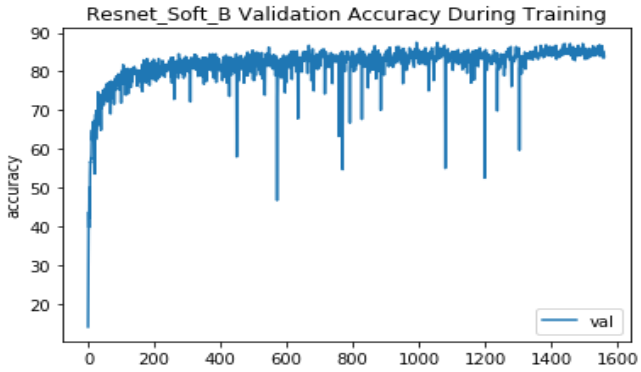
ResNet Models	Accuracy	Previous Models	Accuracy
Resnet (updatable) + softmax	82.96%	SoftMax	40.02%
Resnet (updatable) + 2layer NN	80.75%	2layerNN	51.02%
Resnet (updatable) + convnet	80.91%	Convnet	58.9%
Resnet (updatable) + mymodel	80.07%	Mymodel	76%

Model Parameters:

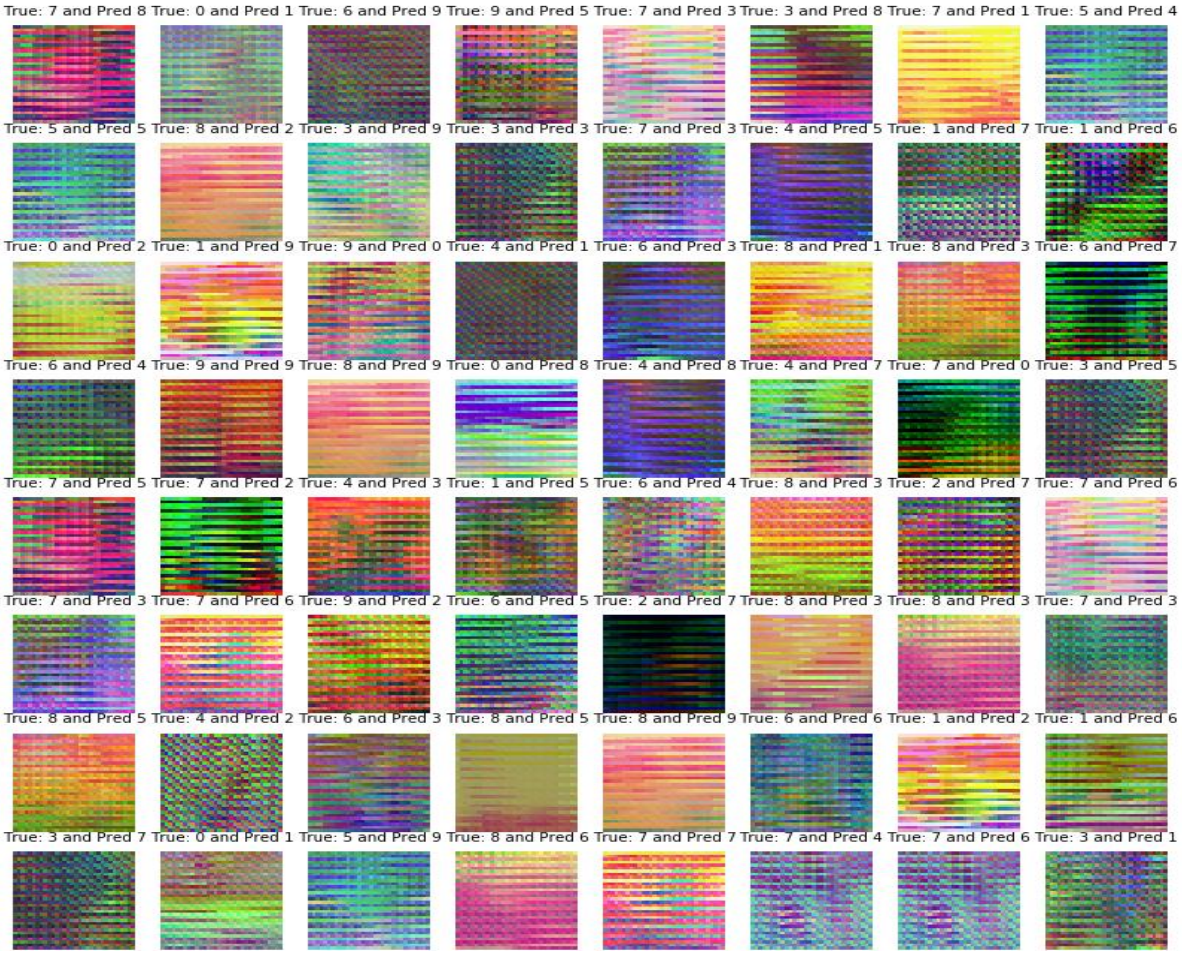
Models	Best Parameters
Resnet (updatable) + Softmax	--epochs 40 --weight-decay 0.001 --momentum 0 --batch-size 128 --lr 0.01
Resnet (updatable) + 2layer NN	--hidden-dim 150 --epochs 30 --weight-decay 0.0 --momentum 0.99 --batch-size 128 --lr 0.001
Resnet (updatable) + Convnet	--kernel-size 3 --hidden-dim 64 --epochs 45 --weight-decay 0.001 --momentum 0.98 --batch-size 128 --lr 0.001
Resnet (updatable) + MyModel	--epochs 50 --hidden-dim 128 --kernel-size 3 --weight-decay 0.0001 --momentum 0.7 --batch-size 128 --lr 0.04

Log visualization plots for ResNet (updatable) + Softmax

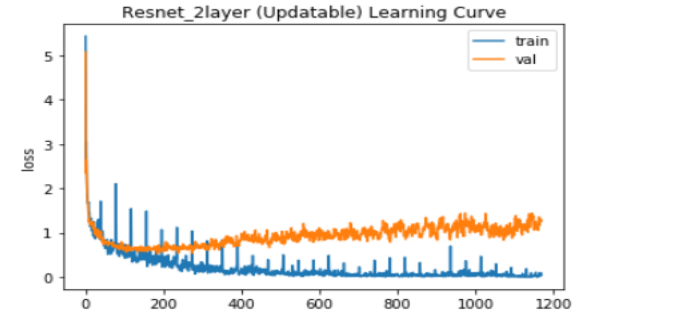


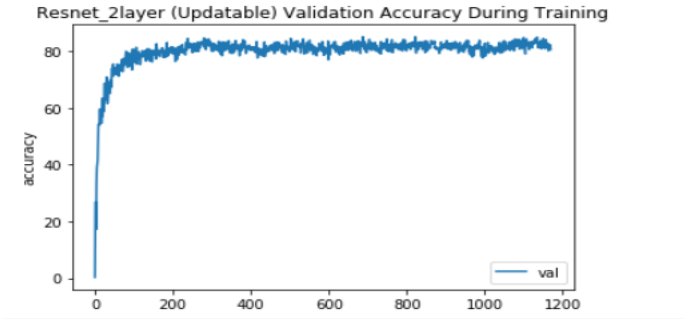


Wrongly classified image

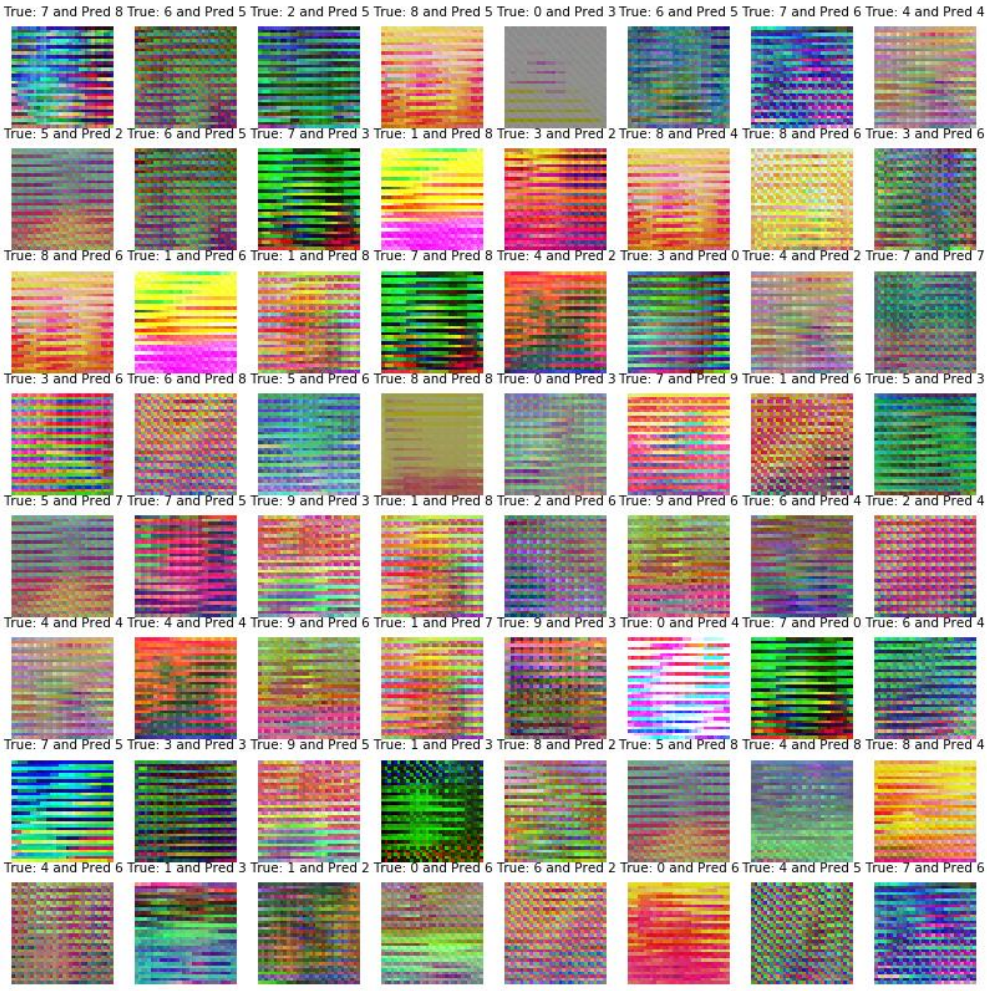


Log Visualization plots for ResNet (updatable) + 2-Layer NN

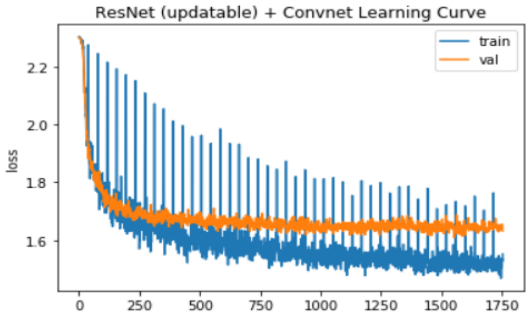


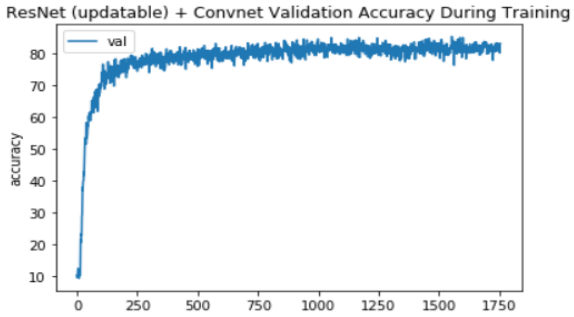


Wrongly classified image

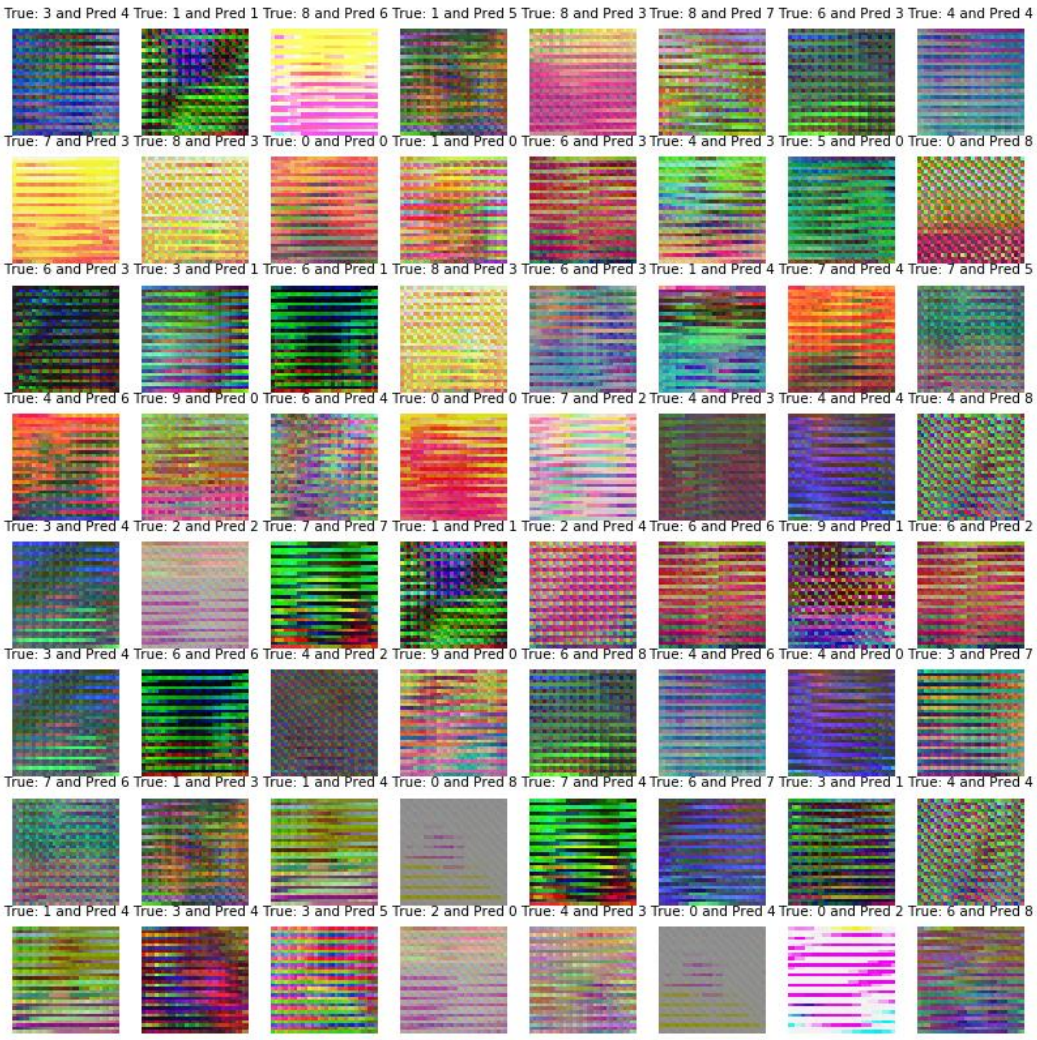


Log Visualization plots for ResNet (updatable) + Convnet

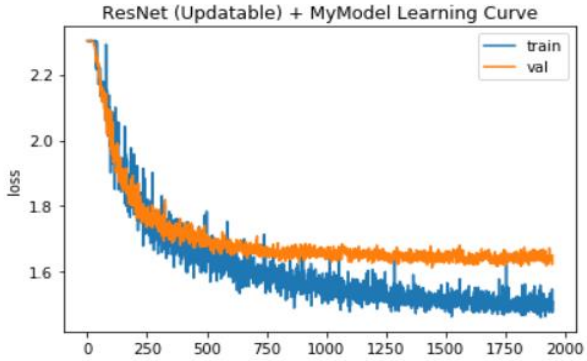


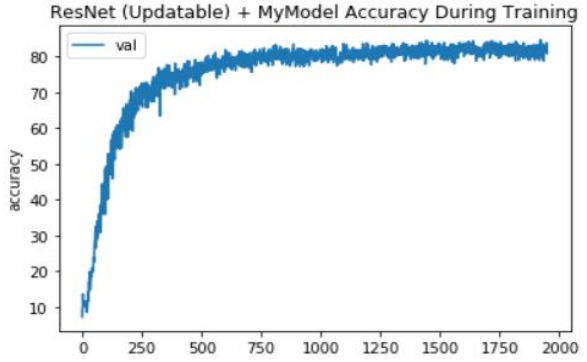


Wrongly classified image

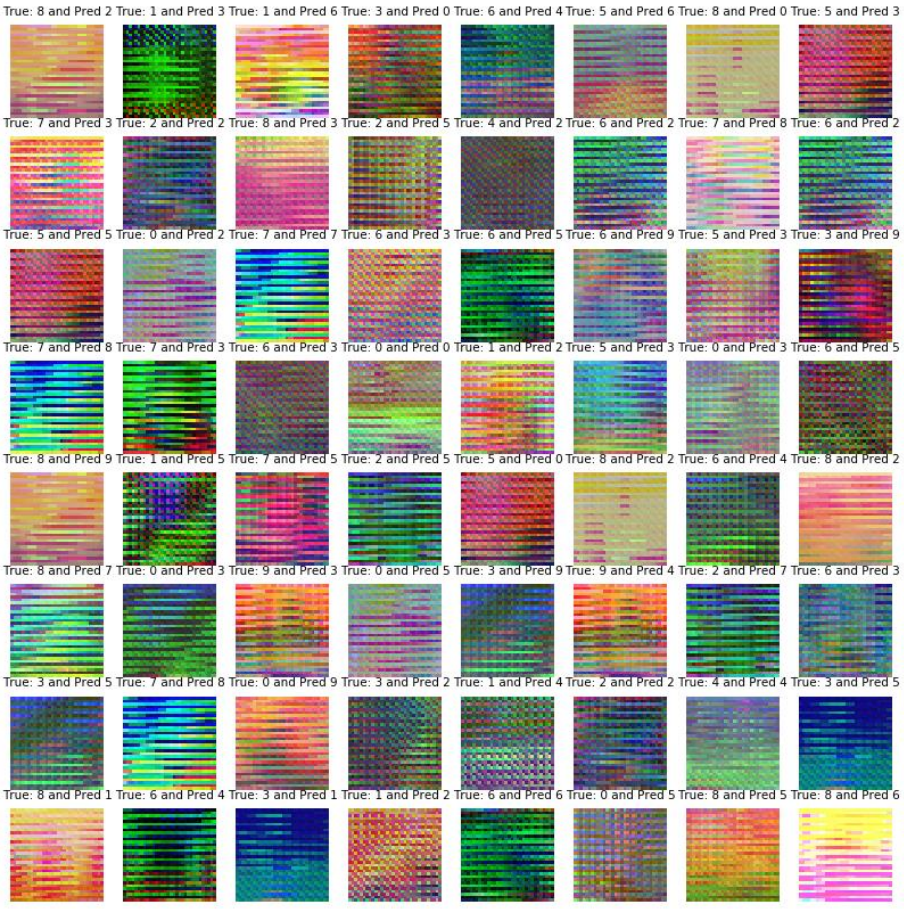


Log Visualization plots for ResNet (updatable) + MyModel



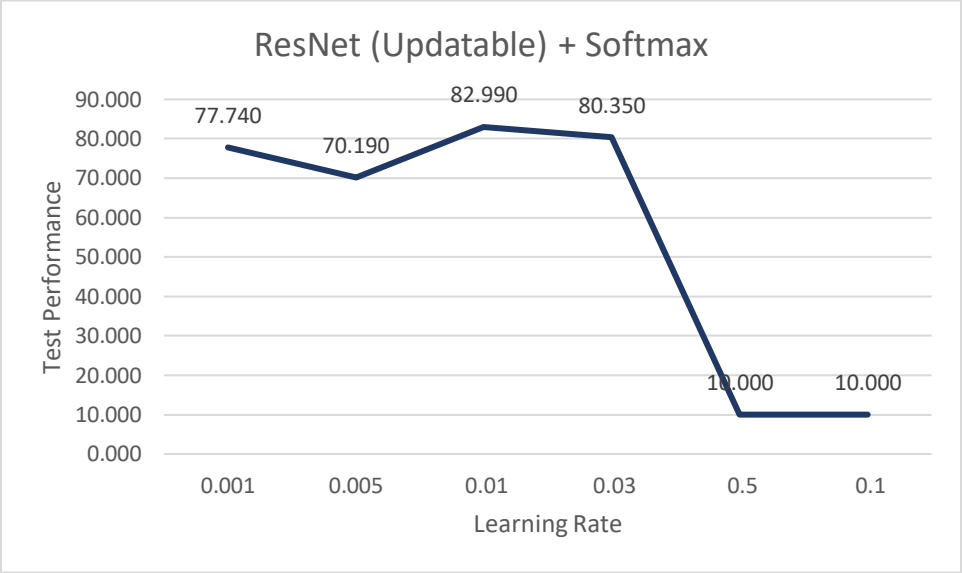


Wrongly classified image

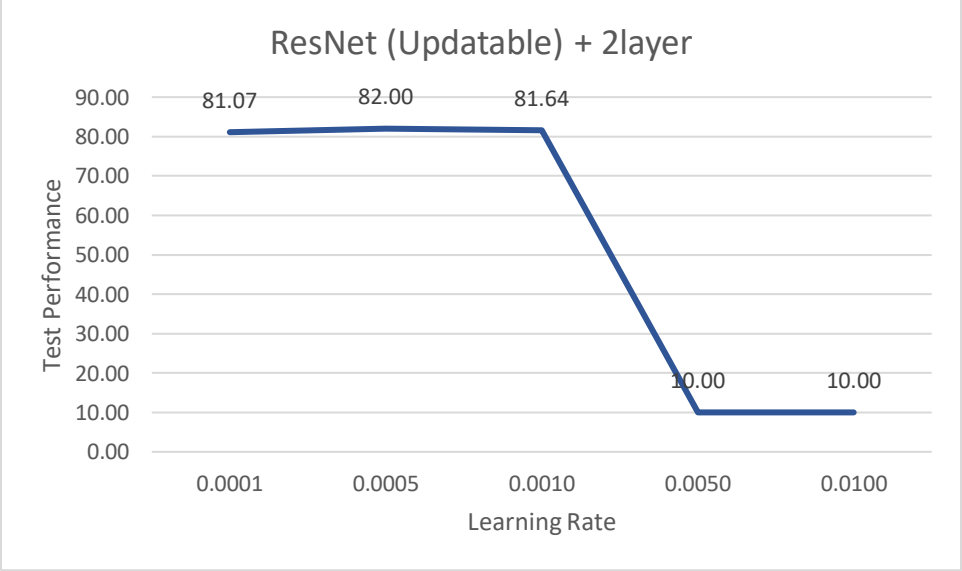


(c) We have performed hyper-parameter tuning in all above sections to identify best parameters and best accuracy for that model. Hence, in this section we are performing learning rate vs accuracy study by keeping all other hyper-parameters equal to one obtained for best accuracy in previous sections (Please refer to model parameters table in answer (b) for all other hyper-parameters than learning rate).

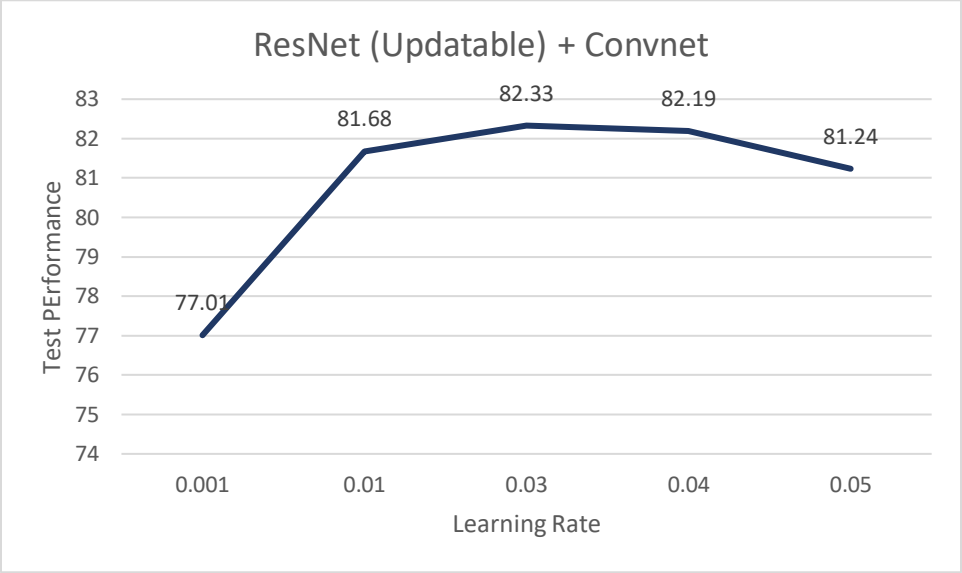
ResNet (updatable) + Softmax



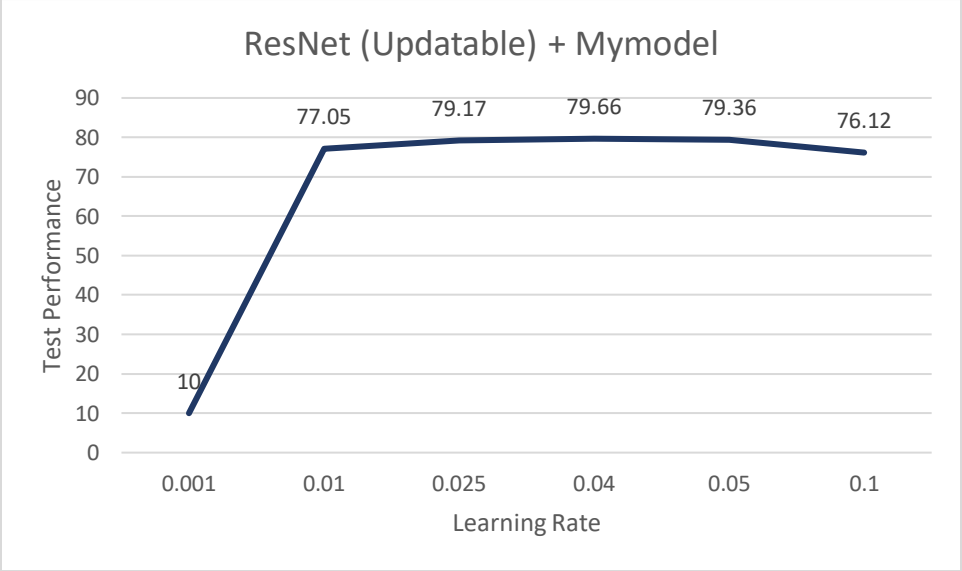
ResNet (updatable) + 2-layer NN



ResNet (updatable) + Convnet



ResNet (updatable) + MyModel

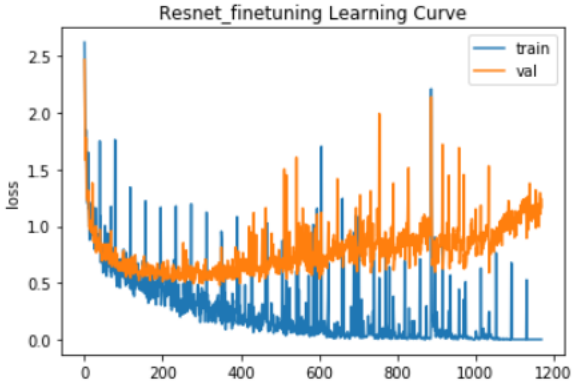


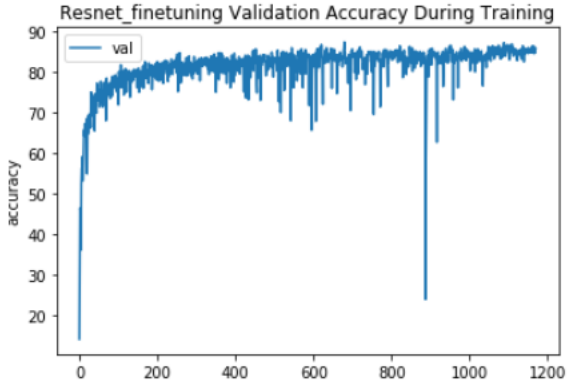
(d) In fine tuning, we are adding one layer further to ResNet-18 architecture. Please refer to script (train_resnet_finetuning.py) for this part.

The best performance is 83.12% with parameters (epochs – 30, weight-decay - 0.001, momentum – 0, batch-size - 128, lr - 0.01). This can further be improved on identifying the best parameters further. The performance of this model is higher than all other models built in previous questions.

Please refer to train_resnet_finetuning.py for the code (Model is built in this file only).

Log Visualization plots for ResNet (updatable) + MyModel





Wrongly classified image