Advanced Predictive Modelling

IDS 576 – ASSIGNMENT 2 REPORT

Question 1: Neural Network Design by PyTorch

a) Does the training use mini-batch (i.e., update the model parameters for each batch of training examples) or update stochastically (i.e., update the model parameters for each example? What is the relation between epoch and batches?

Solution: Training is using mini-batch algorithm. An epoch is when an entire dataset is passed forward and backward through the Neural Network once. As we cannot feed the entire dataset to the neural network, we usually feed out data in sets/blocks also known as batches. For ex. If we have 10000 training examples and the batch size is 5000, then it will take 2 iterations to complete 1 epoch.

b) The first TO-DO is located before defining our training function. Similar to what we have learned in PyTorch examples in class, you are supposed to instantiate at least one optimizer among many predefined options in the torch.optim package. Be careful in receiving the model parameters. (Note: Learning Rate (lr) and Momentum (momentum)2 must be plugged directly from what you have received in the command-line options rather than fixing as single values)

Solution: Initializing the optimizer – Code part of the file – Train.py

```
################################################################################
# TODO: Initialize an optimizer from the torch.optim package using the      #
# appropriate hyperparameters found in args. This only requires one line.   #
################################################################################
#pass

if args.model == 'mymodel':
    #optimizer = optim.Adam(model.parameters(), lr=args.lr,weight_decay=args.weight_decay)#, momentum=args.momentum
    optimizer = optim.SGD(model.parameters(), lr=args.lr, momentum=args.momentum,weight_decay=args.weight_decay)
else:
    optimizer = optim.SGD(model.parameters(), lr=args.lr, momentum=args.momentum,weight_decay=args.weight_decay)

################################################################################
#                            END OF YOUR CODE                                  #
################################################################################
```

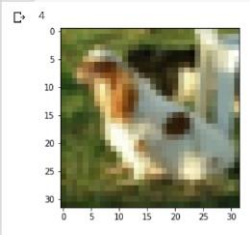
c) Inside the train function, for each batch of training images, you should get the outputs by feeding the current batch of images to the model. Then you should explicitly compute the loss based on the criterion that is already selected as Cross Entropy.3 Note also that given the loss, you should

perform one backward pass, later updating the model parameters. (Note: Be careful to remove existing gradient information before running one backward pass

Solution: Update model parameters – Code part of the file – Train.py

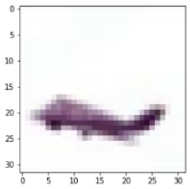
```
################################################################################
# TODO: Update the parameters in model using the optimizer from above.         #
# This only requires a couple lines of code.                                   #
################################################################################
#pass
# can try with reduction=mean too.....
pred=model(images)
loss=criterion(pred,targets)
optimizer.zero_grad()
loss.backward()
optimizer.step()

################################################################################
#                                END OF YOUR CODE                              #
################################################################################
```

Error Analysis – Q2(e)



Predicted Class: Cat

Actual Class: Dog



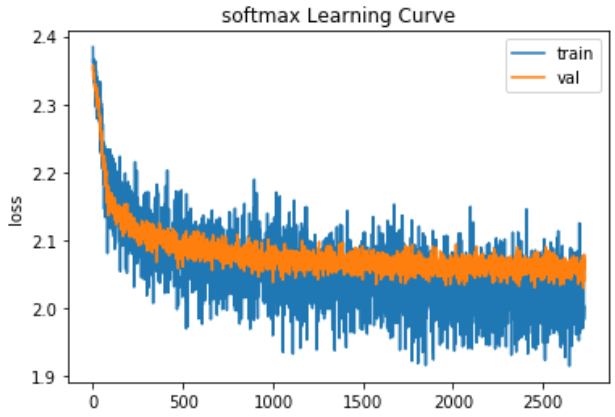
Predicted Class: Bird

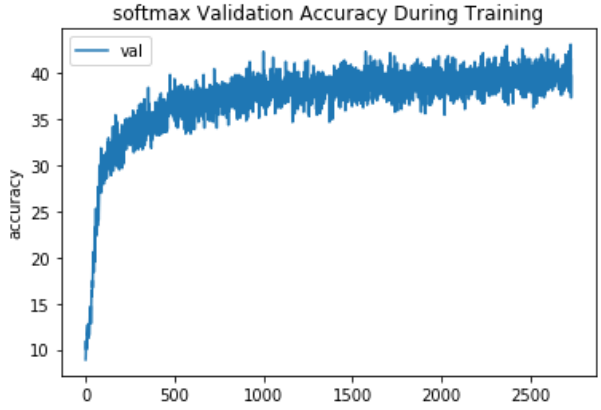
Actual Class: Airplane

There can be various reasons for the images to be misclassified by the CNN – easy images in training, orientation, difference in shapes, aspect ratio, difficult to separate two classes etc. A couple of difficult images have been printed above. First image is of a dog and from this picture we can see that the orientation of the image is such that the dog is not clearly facing the image, hence, this image was misclassified as a cat. The second image above is of a plane and the orientation and shape of the plane is such it is looking like a bird and has been misclassified as a bird instead.

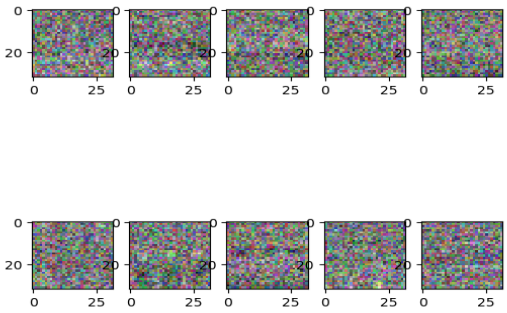
Question 2: Softmax (Multiclass) Classification

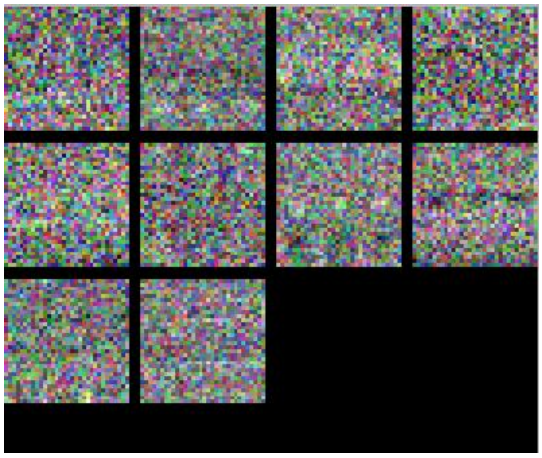
- ✓ *Softmax Learning Curve and Validation Accuracy plots -*





- ✓ *Softmax Best Weight – parameter - Plot*

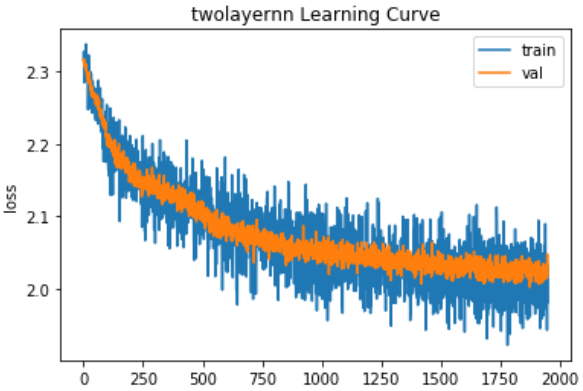


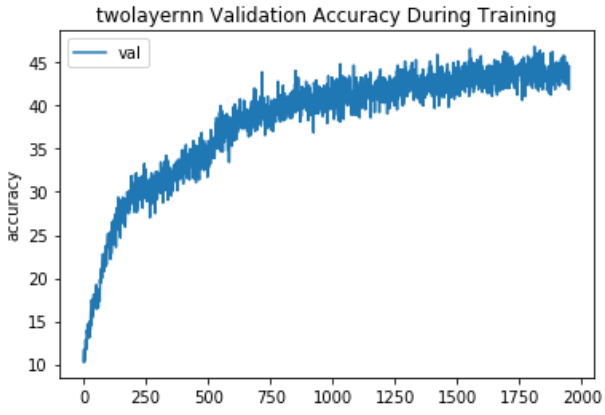


- ✓ Hyperparameters tuned -> epochs, weight-decay, momentum, batch-size, learning rate
- ✓ Overall test Accuracy obtained on the dataset is -> 40%
- ✓ **Refer the Best Parameters and Logs from softmax.log**

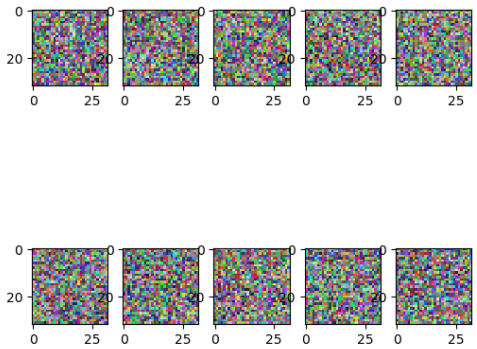
Question 3: Two-layer (Multiclass) Classification

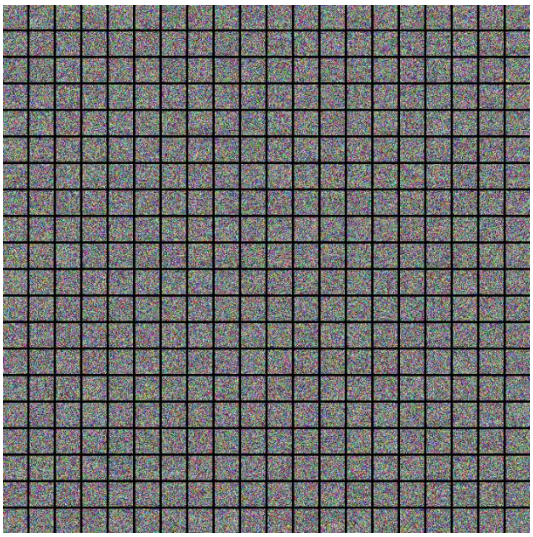
- ✓ *Two-Layer Learning Curve and Validation Accuracy plots -*





- ✓ *Two-Layer Best Weight – parameter – Plot*

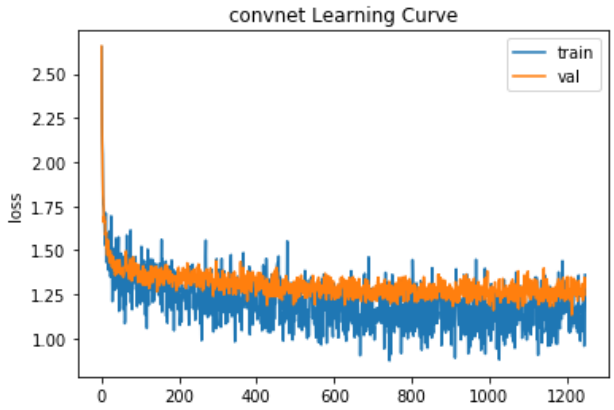


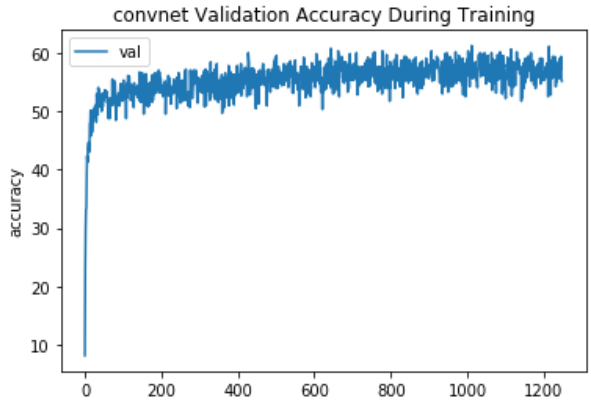


- ✓ Hyperparameters tuned -> hidden, dim., epochs, weight-decay, momentum, batch-size, learning rate
- ✓ Overall test Accuracy obtained on the dataset is -> 43%
- ✓ **Refer the Best Parameters and Logs from twolayer.log**

Question 4: Convolutional Neural Network

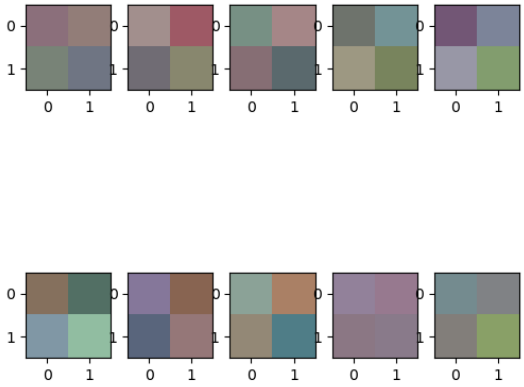
- ✓ *CNN Learning Curve and Validation Accuracy plots -*

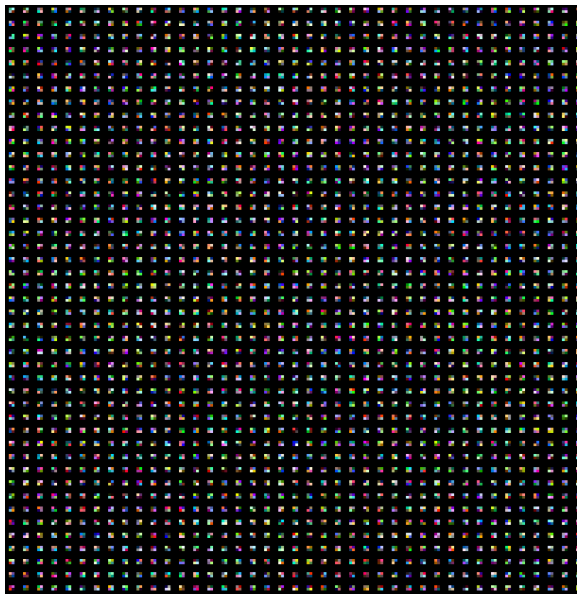




- ✓ *CNN Best Weight – parameter – Plot*

***Kernel Size=2** (can be seen from the figure below)*

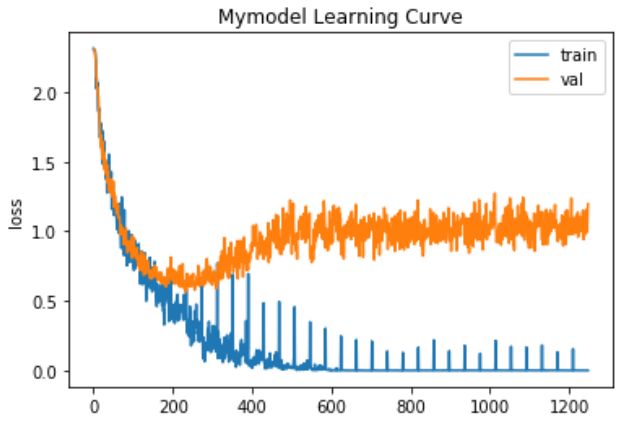


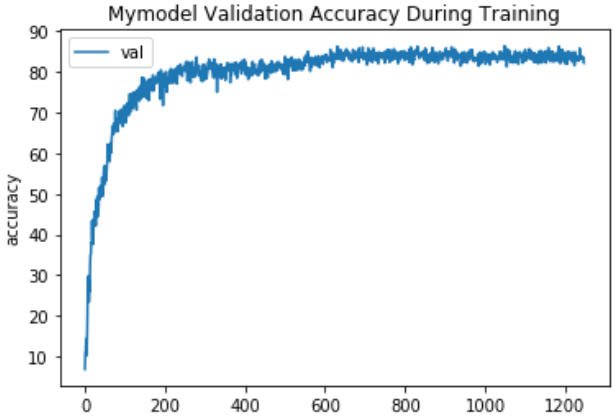


- ✓ Hyperparameters tuned -> Kernel size, hidden, dim., epochs, weight-decay, momentum, batch-size, learning rate
- ✓ Overall test Accuracy obtained on the dataset is -> 56%
- ✓ **Refer the Best Parameters and Logs from convnet.log**

Question 5: MyModel Neural Network

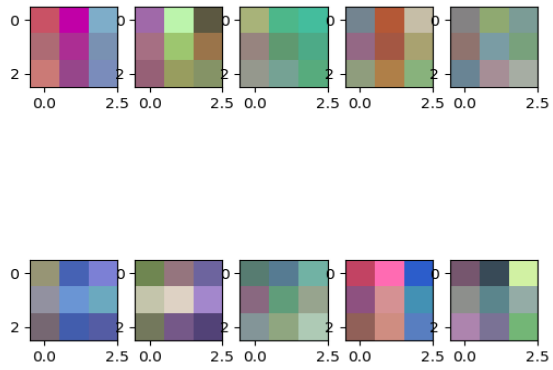
- ✓ *CNN Learning Curve and Validation Accuracy plots -*

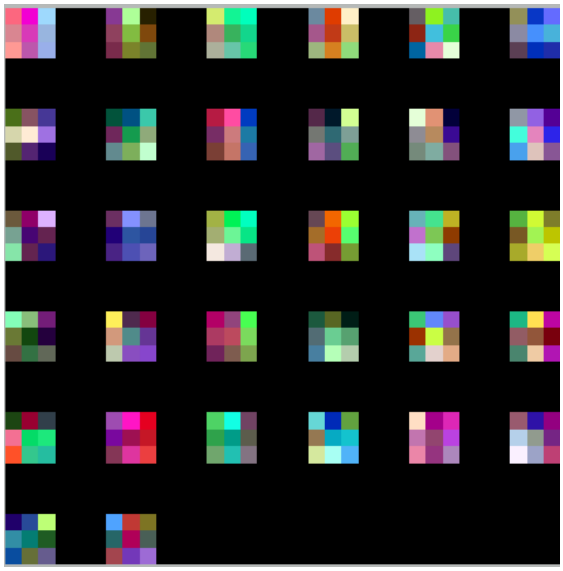




- ✓ *CNN Best Weight – parameter – Plot*

Kernel Size=3*(can be seen from the figure below)*



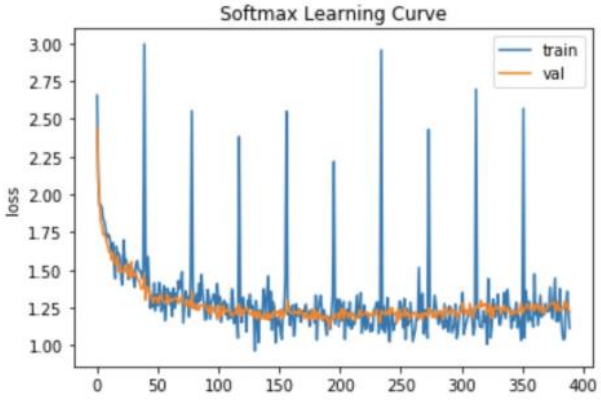


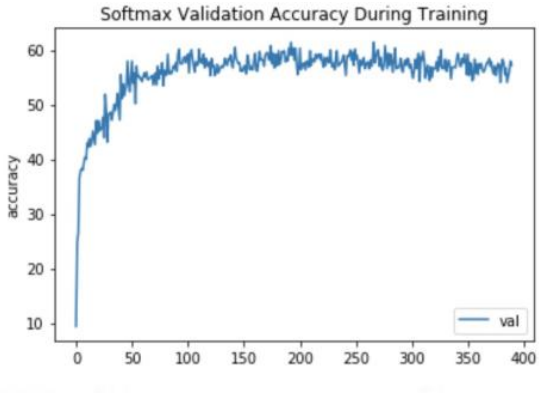
- ✓ Hyperparameters tuned -> Kernel size, hidden, dim., epochs, weight-decay, momentum, batch-size, learning rate
- ✓ Overall test Accuracy obtained on the dataset is -> 81%
- ✓ **Refer the Best Parameters and Logs from mymodel.log**

Problem 6: Transfer Learning

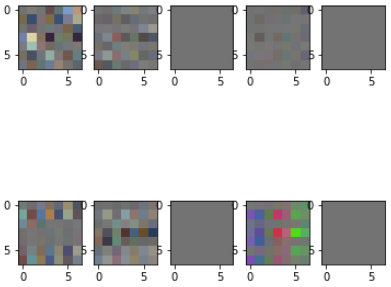
a) Feed each image to the pretrained ResNet-18 nerual network. Extract the CNN-codes which is before the fully-connected layer but after the final ReLU. Use the codes as your initial representations of CIFAR-10 data for the previous four models in Problem 2, 3, 4, 5. You should re-train the model with the CIFAR-10's training data, pick the best hyperparameters with respect to CIFAR-10's validation data. Then report the performance of each model on CIFAR-10's test data comparing to the previous results. Analyze the results.

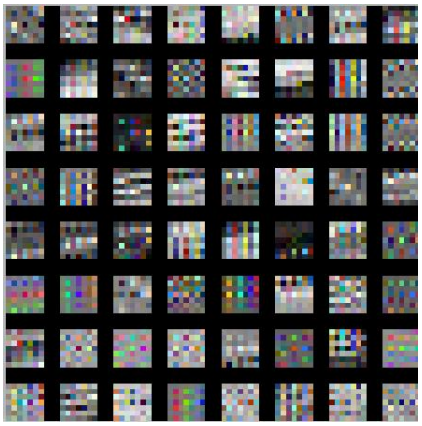
- ✓ *Using trained weights of Resnet -18 i.e. not updating back the parameters(requires_grad=FALSE)*
- ✓ *Softmax using Resnet - Curve and Validation Accuracy plots -*





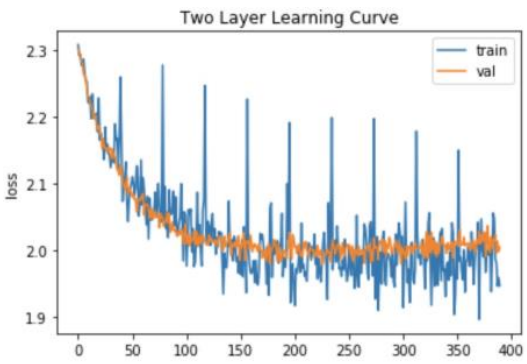
- ✓ *Softmax with Resnet Best Weight – parameter – Plot*

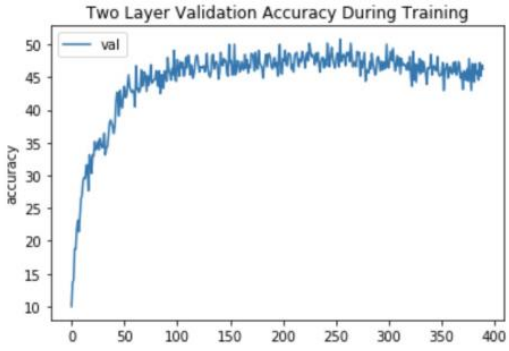




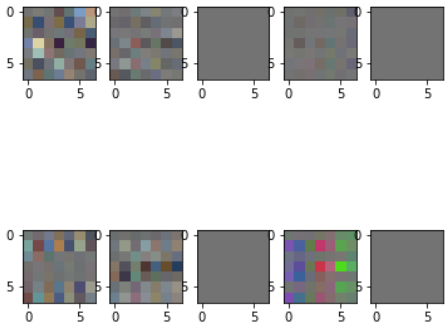
- ✓ Overall test Accuracy obtained on the dataset is -> 56%
- ✓ **Refer the Best Parameters and Logs from softmax_resnet_1.log**

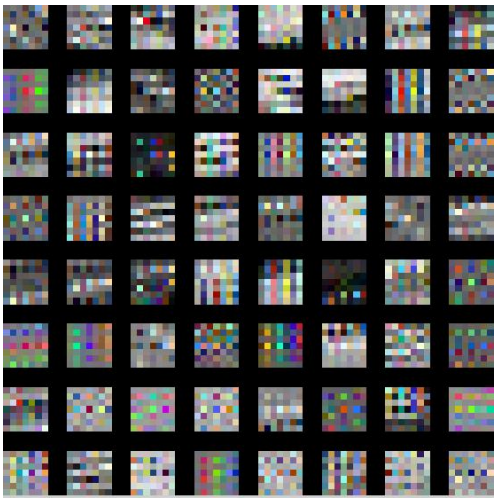
- ✓ *Using trained weights of Resnet -18 i.e. not updating back the parameters(requires_grad=FALSE)*
- ✓ *Two Layer using Resnet - Curve and Validation Accuracy plots -*





- ✓ *Two Layer with Resnet Best Weight – parameter – Plot*

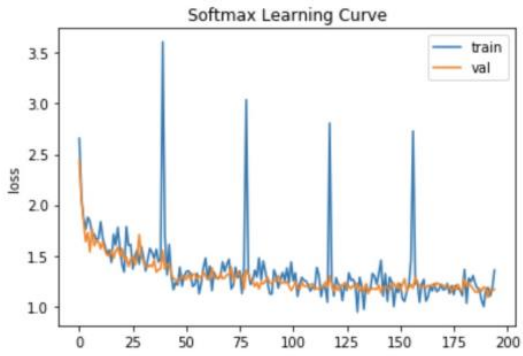


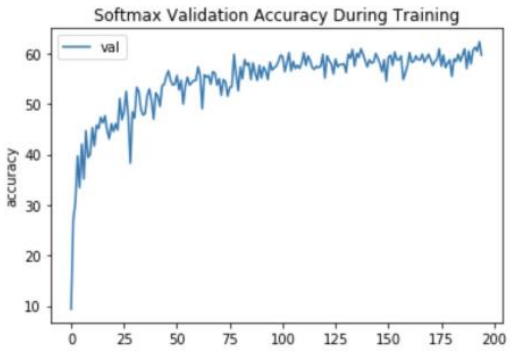


- ✓ Overall test Accuracy obtained on the dataset is -> 48% (Higher accuracy can be achieved for higher number of epochs)
- ✓ **Refer the Best Parameters and Logs from twolayernn_resnet_1.log**

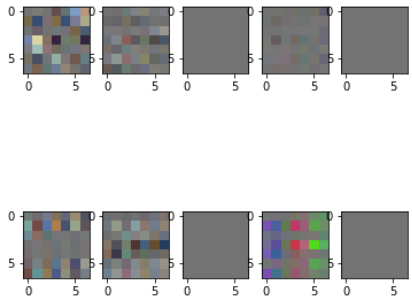
b) **Stay within the setting of (a), but now set the initial representations (i.e., the CNN-codes from the pretrained Resnet-18) as updatable parameters. Retry training, validation, and test on CIFAR-10, then report the test performance of each model. Compare the results to (a)**

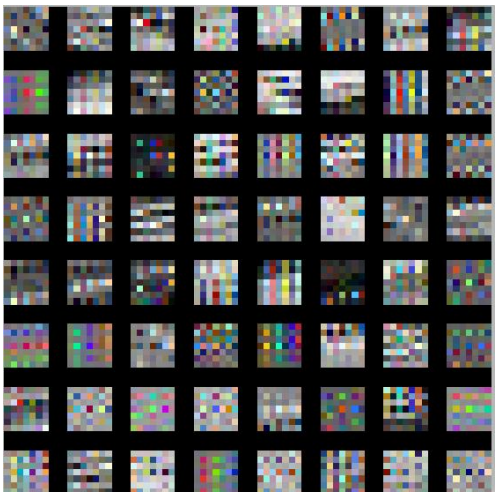
- ✓ *Using trained weights of Resnet -18 and updating parameters only for the fc.layer i.e. (fc.requires_grad=TRUE)*
- ✓ *Softmax using Resnet - Curve and Validation Accuracy plots –*





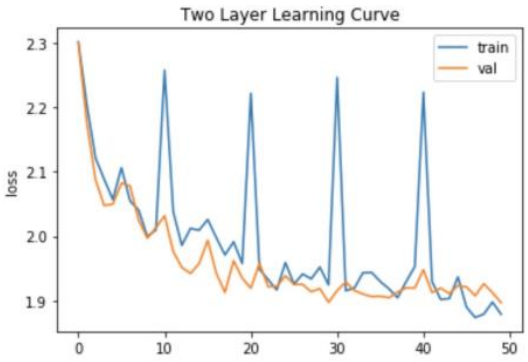
- ✓ *Softmax using Resnet Best Weight – parameter – Plot –*

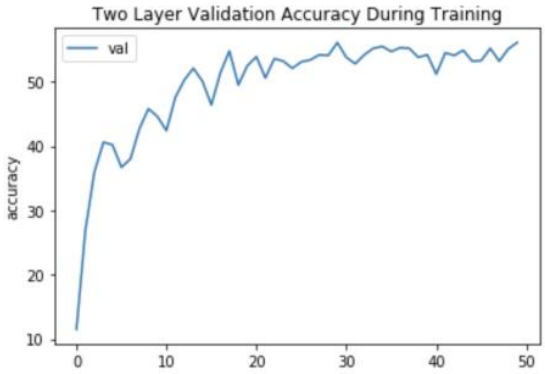




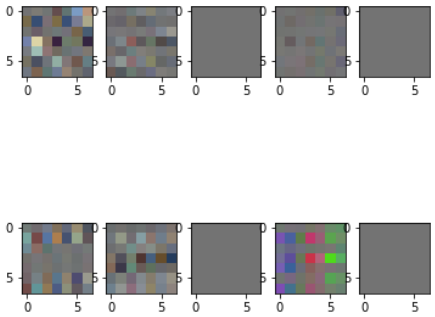
- ✓ Overall test Accuracy obtained on the dataset is -> 58%
- ✓ **Refer the Best Parameters and Logs from softmax_resnet_2.log**

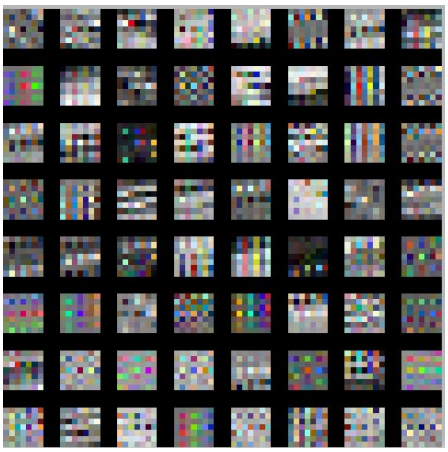
- ✓ *Using trained weights of Resnet -18 and updating parameters only for the fc.layer i.e. (fc.requires_grad=TRUE)*
- ✓ *Two Layer using Resnet - Curve and Validation Accuracy plots –*





- ✓ *Two Layer with Resnet Best Weight – parameter – Plot*

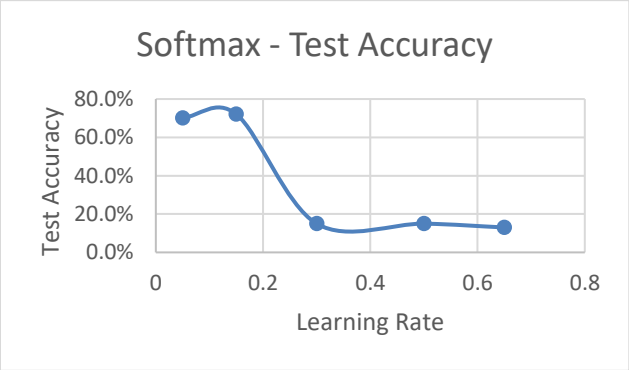




- ✓ Overall test Accuracy obtained on the dataset is -> 52% (Higher accuracy can be achieved for higher number of epochs)
- ✓ **Refer the Best Parameters and Logs from twolayernn_resnet_2.log**

c) **Keep trying (b) with different learning rate. Draw a graph where x-axis is the learning and y-axis is the test performance. Find the best learning rate for each of your four models. Compare the test performance with the best learning rate against your previous best results at (b)**

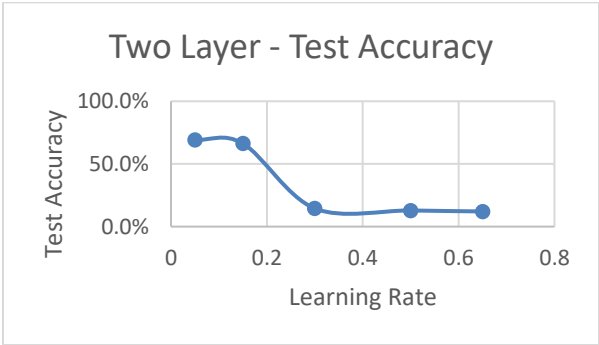
Softmax	
LearningRate	**TestAccuracy**
0.05	70.2%
0.15	72.1%
0.3	15.0%
0.5	15.0%
0.65	13.0%



** Accuracy value will be more if more epochs are used

** of epochs: 1

TwoLayer	
LearningRate	**TestAccuracy**
0.05	69.3%
0.15	66.3%
0.3	14.5%
0.5	12.9%
0.65	12.1%

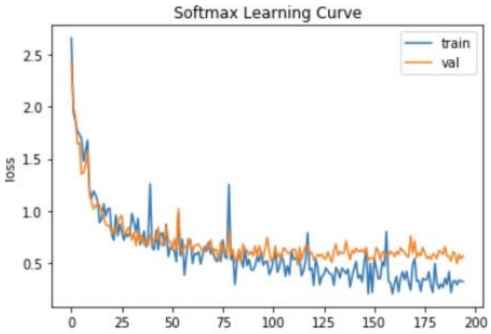


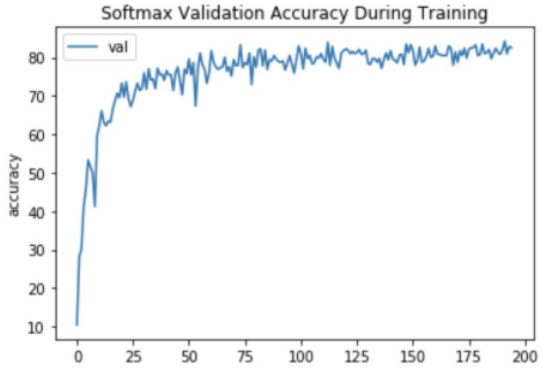
** Accuracy value will be more if more epochs are used

** of epochs: 1

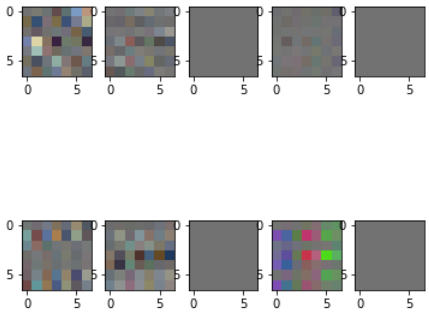
d) Rather than simply reusing the CNN-codes as initial representations (fixed manner at (b), updatable manner at (c)), now you are to finetune the entire ResNet-18 network by feeding CIFAR-10's training data and backpropagation. Pick the best hyperparameters for this finetuning by using CIFAR-10's validation data. Report the test performance with the best hyperparameters against your previous best results at (c)

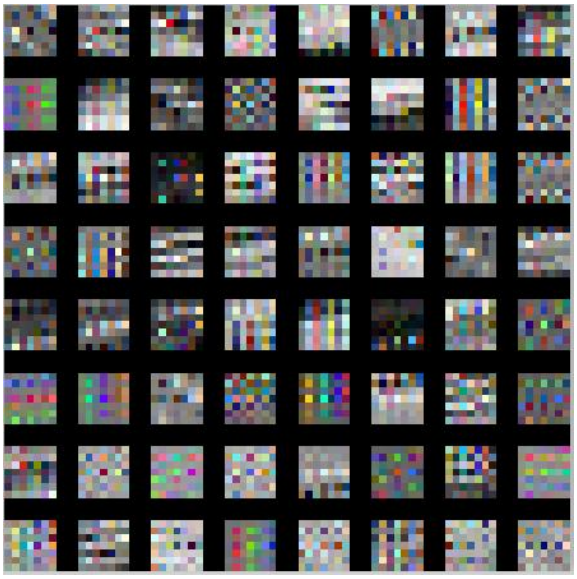
- ✓ *Using trained weights of Resnet -18 and updating all the parameters i.e. backpropagating to the first Resnet layer*
- ✓ *Softmax using Resnet - Curve and Validation Accuracy plots –*





- ✓ *Softmax using Resnet Best Weight – parameter – Plot –*

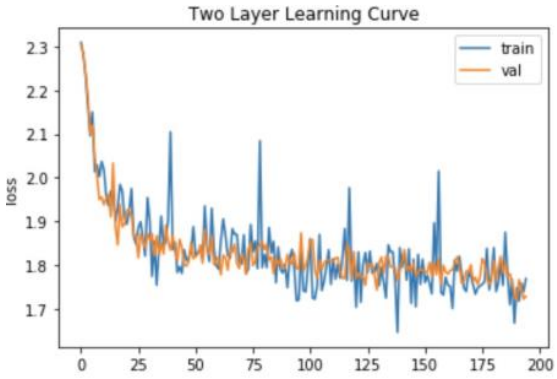


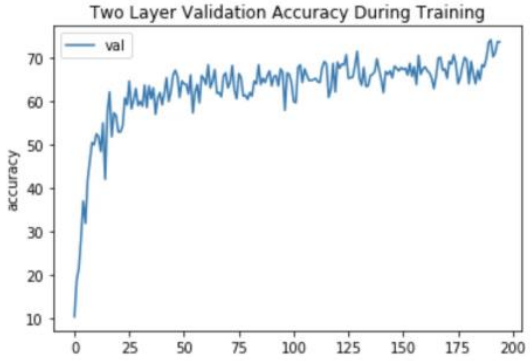


- ✓ Overall test Accuracy obtained on the dataset is -> 82%
- ✓ **Refer the Best Parameters and Logs from softmax_resnet_3.log**

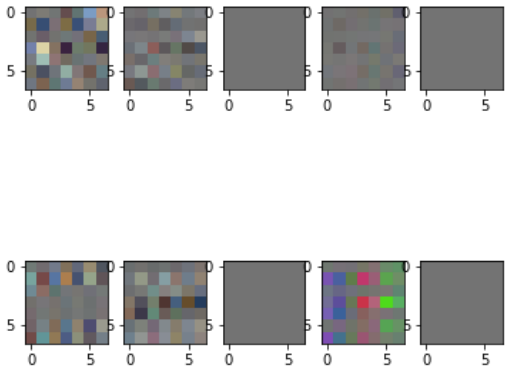
- ✓ *Using trained weights of Resnet -18 and updating all the parameters i.e. backpropagating to the first Resnet layer*

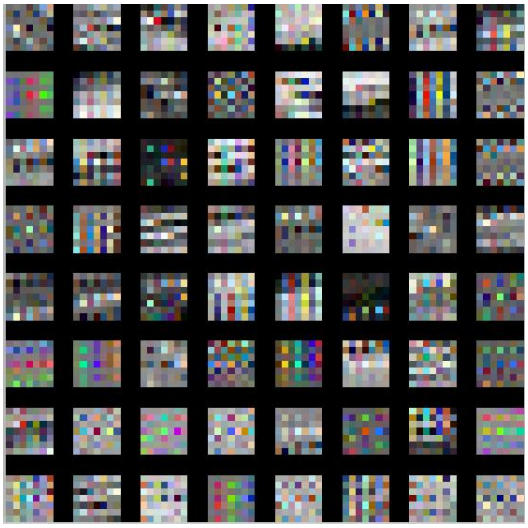
✓ *Two Layer using Resnet - Curve and Validation Accuracy plots –*





✓ *Two Layer with Resnet Best Weight – parameter – Plot*





✓ Overall test Accuracy obtained on the dataset -> 72% (Higher accuracy can be achieved for higher number of epochs)

✓ **Refer the Best Parameters and Logs from twolayernn_resnet_3.log**