Deniz Aytemiz
denizaytemiz@vt.edu

ECE/CS 6524- Deep Learning- HW4- Multilayer Perceptrons

Question 1.

For cifar10 dataset the model used is as represented as below,
Inputs are scaled to range [0,1] and they are reshaped to (batch_size, 3072) where for each image the array of size 3072 will present as 32x32 pixel image for RGB resulting in 32x32x3 = 3072.
Labels are one hot encoded hence they are in shape (batch_size,100).
Therefore the output layer has 10 nodes each holds a score on the likeliness of the label belonging to input is that particular node.

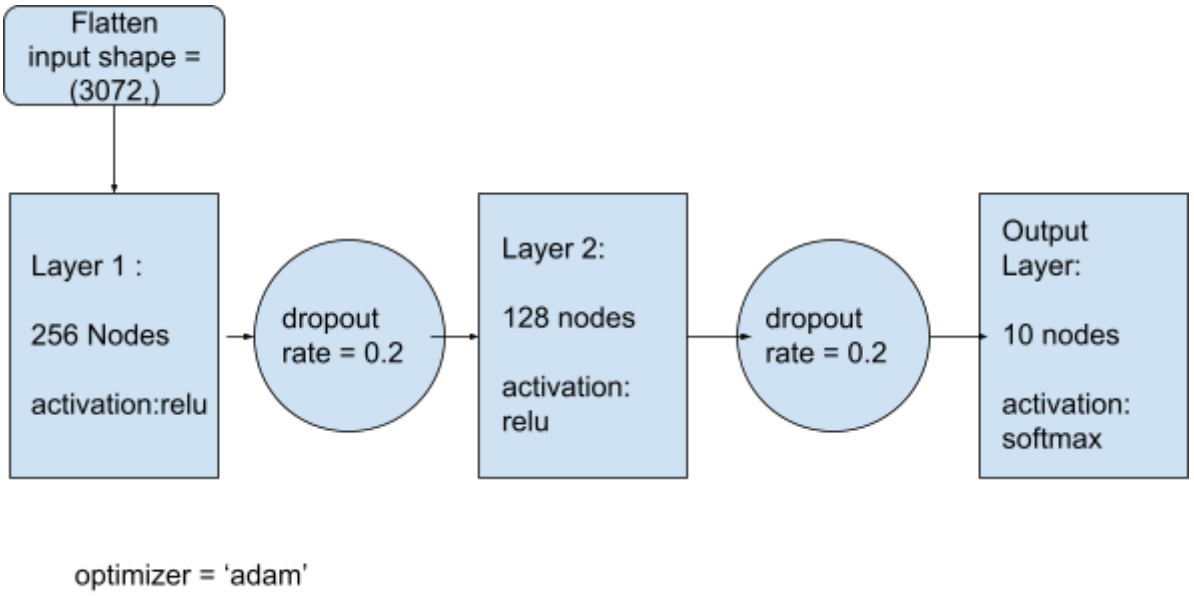


Figure 1

For cifar100 dataset the model used is as represented as below,
Inputs are scaled to range [0,1] and they are reshaped to (batch_size, 3072) where for each image the array of size 3072 will present as 32x32 pixel image for RGB resulting in 32x32x3 = 3072.
Labels are one hot encoded hence they are in shape (batch_size,100).
Therefore the output layer has 100 nodes each holds a score on the likeliness of the label belonging to input is that particular node.

Deniz Aytemiz
denizaytemiz@vt.edu

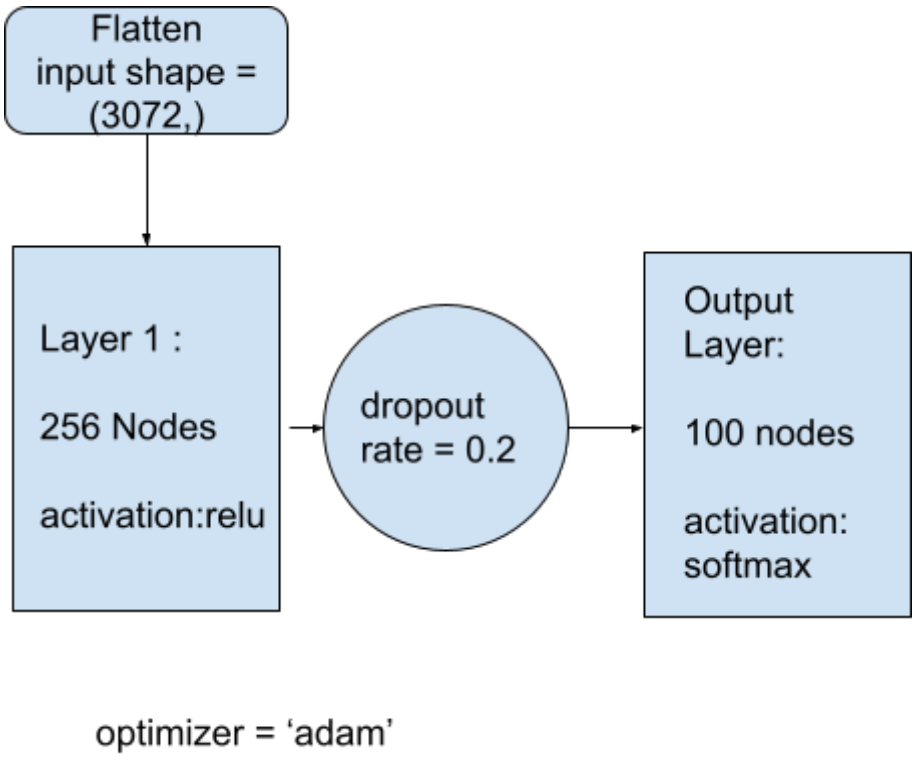


Figure 2

Experimental Results:
For cifar10 dataset the test accuracy I got is 0.4318999946117401
For cifar100 dataset the test accuracy I got is 0.12319999933242798.

Discussion:
The low test accuries may be due to the simplicity of the model. For image classification CNN is better since it has a process of feature selection, whereas with the models used in this assignment have no feature selection. Also they dont have many hidden layers. Another thing can be the different choice of hyperparameters. However I have tried grid search for the model for cifar10 dataset and did not achieve much better results in terms of test accuracy even with the variety of parameters.

Question 2.
Experimental Results: Variety of models are trained and tested. For the model proposed in Question1 in Figure 1, its performance with variations in that model are recorded and summarized in the table below.

model	model in Figure1	model with dropout rate 0.6 (model2 in code)	model without dropout (model4 in code)	model with optimizer SGD	model with no dropout and SGD as optimizer
test accuracies	0.4154999852180481	0.2687000036239624	0.42179998755455017	0.42309999465942383	0.43149998784065247

Table 1

Deniz Aytemiz
denizaytemiz@vt.edu

For the same dataset I have used another model with different structure, it is represented in below,
model 6 :

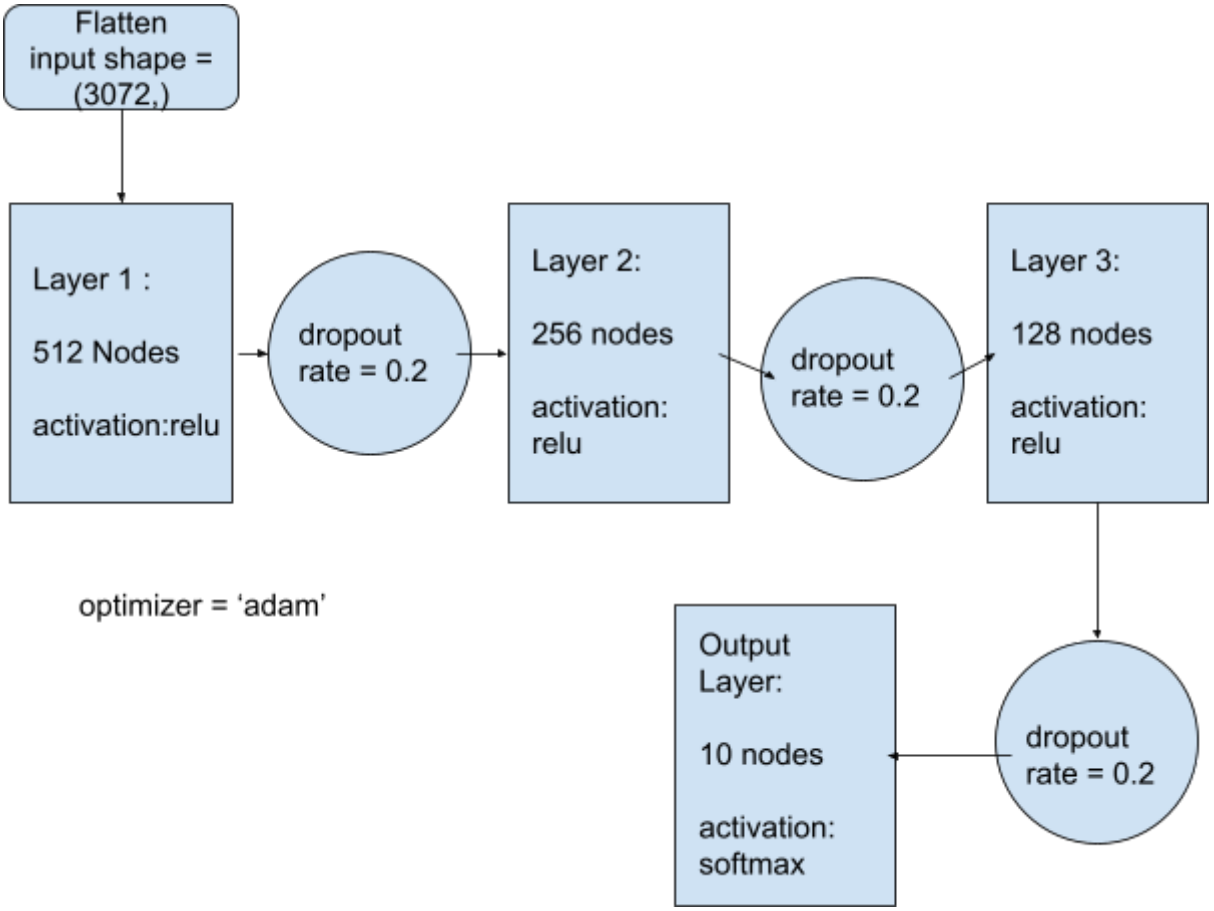


Figure 3

The test accuracy it gives is 0.4487999975681305.

For the variations of model in Figure 2 used for dataset cifar100, the test accuracies are summarized in below,

model	model in Figure 2	model with dropout rate 0.6 (model3 in code)	model without dropout (model5 in code)	model with optimizer SGD	model with no dropout and SGD as optimizer
test accuracies	0.1270000040 5311584	0.0540999993 68190765	0.1469999998 807907	0.1301999986 1717224	0.1588000059 1278076

Table 2

In addition for cifar100 dataset, I have used another model with more layers represented in below,

Deniz Aytemiz
denizaytemiz@vt.edu

model7 =

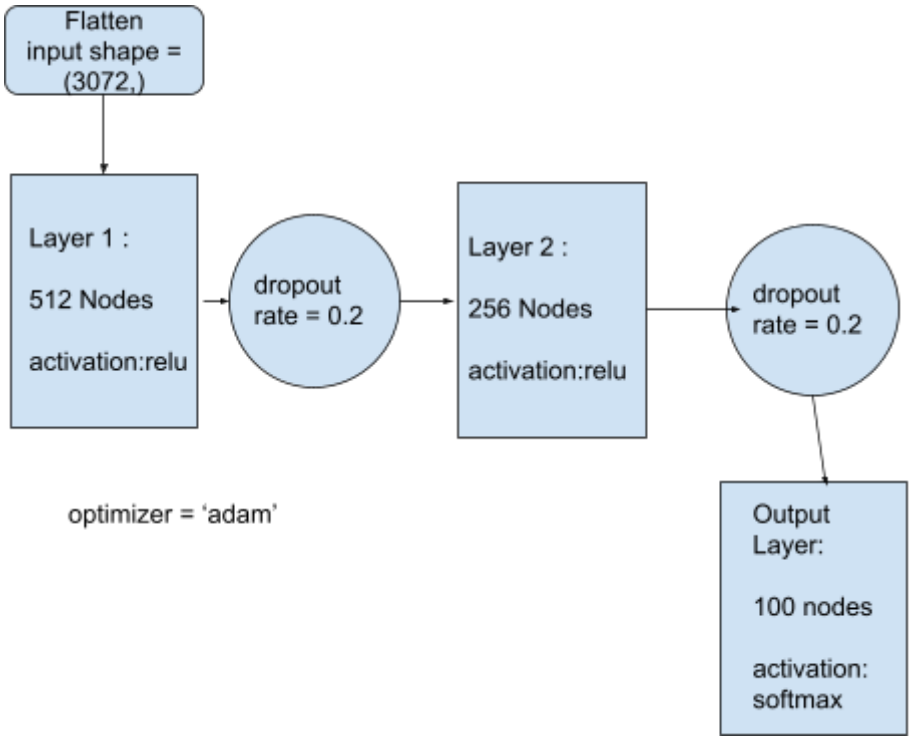


Figure 4

This gave test accuracy of 0.16419999301433563.

Discussion: For the models used for cifar10 dataset, the best performance is given by model 6 which has more layers then the model in Figure 1. In addition among the variations of the model in Figure 1, it can be observed that increasing dropout decreases test accuracies so, the model requires more nodes and probably more layers too since model 6 gives higher accuracy. Also it can be observed that SGD as optimizer seems to be performing better than optimizer adam for this model. Therefore among all the variations of the model in Figure 1 the best performing one in no dropout and SGD as optimizer.

For the models used for cifar100 dataset, the best performance is given by model 7 which has more layers then the model in Figure 2. In addition among the variations of the model in Figure 2, it can be observed that increasing dropout decreases test accuracies so, the model requires more nodes and probably more layers too since model 7 gives higher accuracy. Also it can be observed that SGD as optimizer seems to be performing better than optimizer adam for this model too. Therefore among all the variations of the model in Figure 2 the best performing one in no dropout and SGD as optimizer.

Question 3.
For this part I have builded a model with 2 layers and no hidden layer for dataset cifar10, where the number of nodes are 3072 and 10 respectively. I tried to get the weight array for each layer and visualize them for some insight in feature extraction.
The model properties can be observed in Figure 5

Deniz Aytemiz
denizaytemiz@vt.edu

```
model8.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (2000, 3072)              9440256

 dense_1 (Dense)             (2000, 10)                30730

=================================================================
Total params: 9,470,986
Trainable params: 9,470,986
Non-trainable params: 0
_________________________________________________________________
```

Figure 5

Experimental Results: The weights of output layer are shown in Figure 6.

```
model8.layers[1].weights[0]
```

```
<tf.Variable 'dense_1/kernel:0' shape=(3072, 10) dtype=float32, numpy=
array([[-0.00550829,  0.01719033,  0.01097791, ..., -0.02755688,
         0.04484568,  0.02614558],
       [ 0.00265643,  0.0242255 ,  0.0030675 , ...,  0.00083874,
        -0.00077926,  0.03827155],
       [-0.00661922, -0.02270613, -0.04311538, ..., -0.02310299,
         0.02064343,  0.0348314 ],
       ...,
       [-0.03515977, -0.01436821, -0.03987224, ..., -0.02814916,
        -0.02055985, -0.03020417],
       [-0.00178861, -0.02051111,  0.02896741, ..., -0.02590564,
         0.01658285,  0.04067552],
       [-0.00362039,  0.02543324, -0.01233937, ..., -0.01925258,
         0.02262645, -0.01355581]], dtype=float32)>
```

Figure 6

The weights return trainable and non trainable weights hence weights[0] is the trainable weights of the output layer.

The weights array has shape (3072,10) which I assume it is in the form where elements are wij and i is the index of nodes of previous layer ranging up to 3072 and j is the index of nodes of last layer which ranges up to 10. Therefore I take its transpose and get the first row of the transpose matrix which gives me all the weights related with second output node. That array has size of 3072. I resized it to (32,32x3) in order to plot image of weights and the result is shown in Figure 7.

Deniz Aytemiz
denizaytemiz@vt.edu

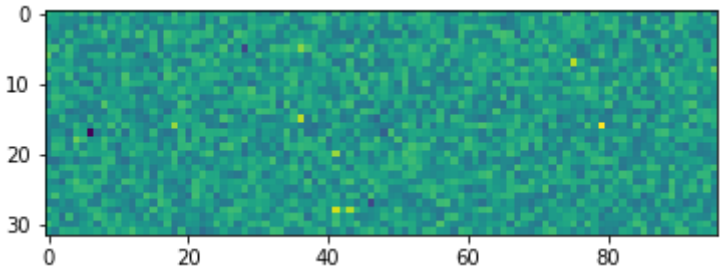


Figure 7

The input layers weight is visualized in Figure 8

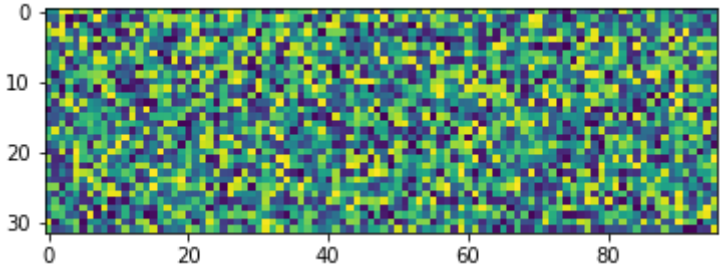


Figure 8

Discussion: Figure 7 shows the weights in 1st layer associated with the 2nd node of the output layer. Figure 8 shows the weights associated with the 2nd node of the output layer. Input layer has more variations of weight which is reasonable since there is no classification there however for the second node of last layer, one can observe almost all weights are close to each other expect from some particular ones has higher weights, which can be indication about the class associated with that node.

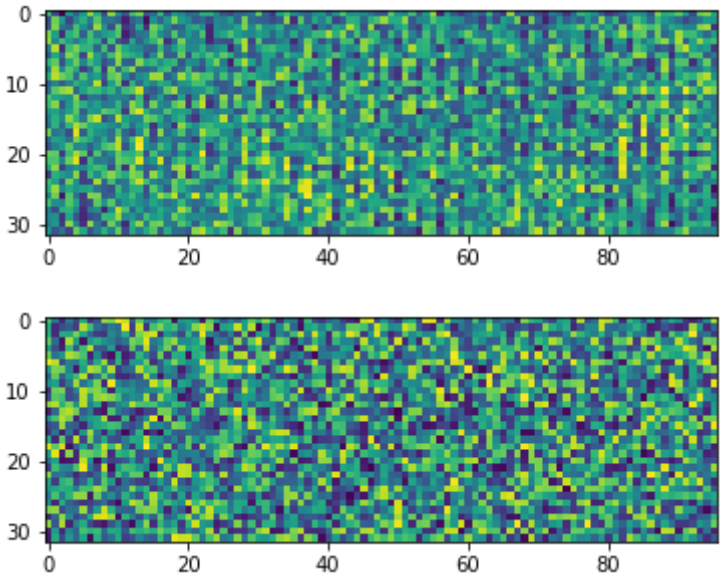
Deniz Aytemiz
denizaytemiz@vt.edu

I modified the model and added 1 hidden layer as given in Figure 9.

```
model8.summary()

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense_2 (Dense)              (2000, 3072)              9440256

 dense_3 (Dense)              (2000, 128)               393344

 dense_4 (Dense)              (2000, 10)                1290

=================================================================
Total params: 9,834,890
Trainable params: 9,834,890
Non-trainable params: 0
_________________________________________________________________
```

Figure 9

The layers weights as visuals is given in Figure 10 for first,hidden and output layer respectively.



Deniz Aytemiz
denizaytemiz@vt.edu

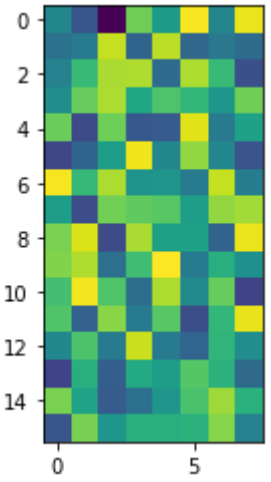


Figure 10

Discussion: It can be observed that he reason first layer weights visual is more uniform looking is that the weights are very close to each other as value which makes sense that they don't differ as much as from each other with respect to hidden layer since it carries more information. In the weights of the hidden layer there is more variation, which gives some insight on the significance of some pixels that could help with classification of that image, as it gives important information.
I think if the number of backpropogation and feed forward increases the weights will differ more from each other and give more insightful visuals overall.