Deniz Aytemiz
denizaytemiz@vt.edu

Project Assignment 1- Convolutional Neural Networks

Part 1: This part discusses experimental results and discussions for cifar10 dataset.
2. In this part of the assignment, the 2 CNN layers are build and the output of these CNN modules are fed into a fully connected layer that outputs the classification of the image fed to the system.
The first model evaluated has the form depicted in Figure 1.

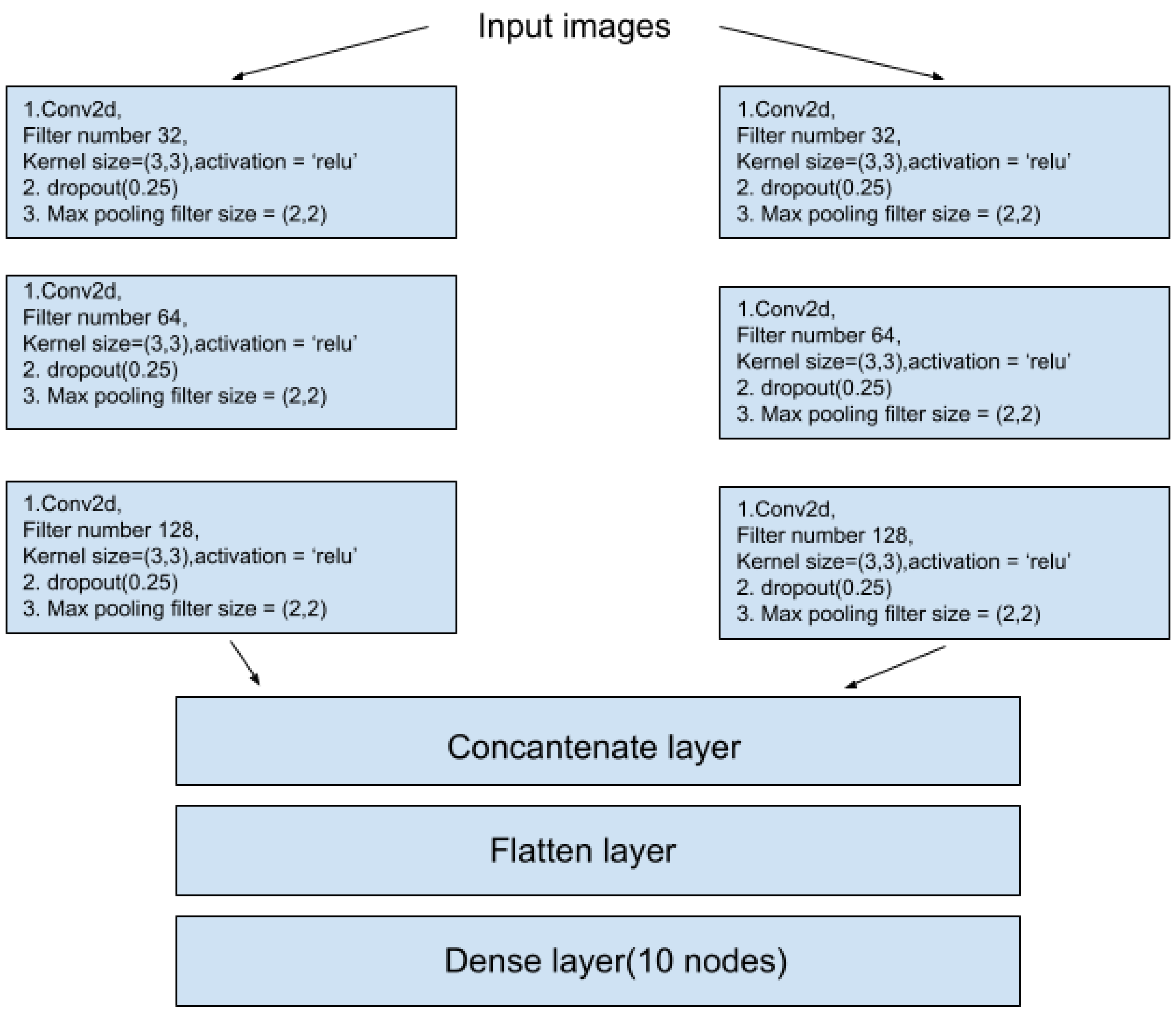

Input images
1.Conv2d,
Filter number 32,
Kernel size=(3,3),activation = 'relu'
2. dropout(0.25)
3. Max pooling filter size = (2,2)
1.Conv2d,
Filter number 64,
Kernel size=(3,3),activation = 'relu'
2. dropout(0.25)
3. Max pooling filter size = (2,2)
1.Conv2d,
Filter number 128,
Kernel size=(3,3),activation = 'relu'
2. dropout(0.25)
3. Max pooling filter size = (2,2)
Concantenate layer
Flatten layer
Dense layer(10 nodes)

Figure 1

This model is trained as, `model.fit(trainX, trainy, batch_size=100, epochs=10, validation_data=(testX, testy))`

And the training and test accuries in each epoch are given in Figure 2.

Deniz Aytemiz
denizaytemiz@vt.edu

Experimental Results:

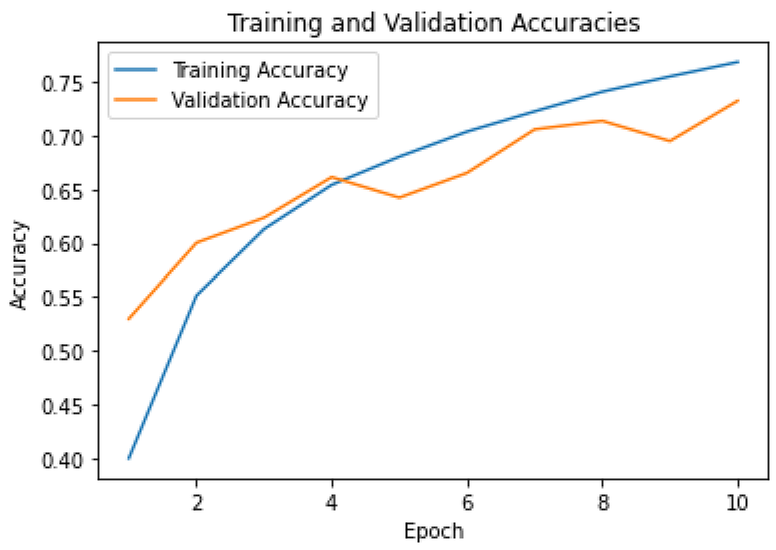


Figure 2

Figure 2 indicates the performance of the model with epoch is set to 10. Overall test accuracy of this model is 0.732200026512146.

The same model trained with 20 epochs and its training and validation accuracies are depicted in Figure 3;

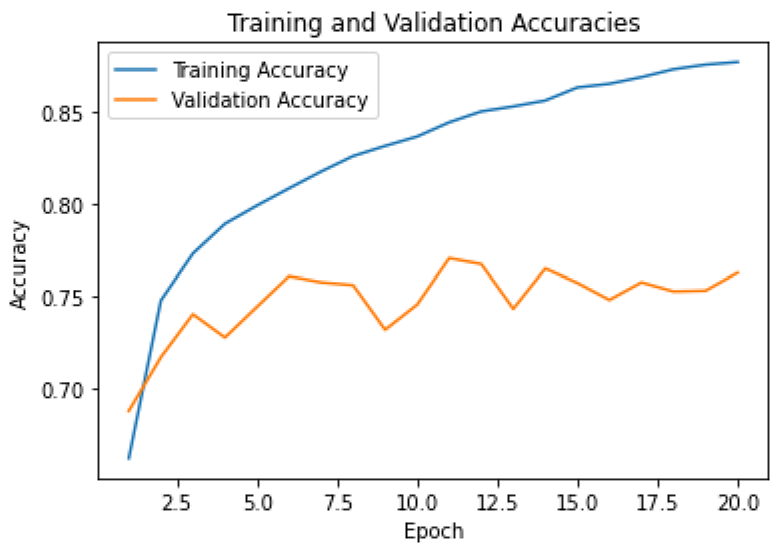


Figure 3

Overall the test accuracy is approximately 0.75.

Deniz Aytemiz
denizaytemiz@vt.edu

Discussion: As seen in Figure 2, there is no overfitting as in each epoch both test and training accuracies increase. If model was overfitting we would expect the training accuracy to increase but test accuracy to decrease. In Figure 3 it can be observed that after epoch 12, validation accuracy does not change much and ripples in a narrow range, whereas the training accuracy still increases. Hence we can say model starts to overfit after this many iterations. It becomes more sensitive to noise and the dataset and its generalization property decreases.

3. I have tried different models with different hyperparameters set. At this part, a grid search can be run and it is present in my code. However it takes a lot of time to train so I have experimented with different hyperparameters by changing the create_cnn_model() function and trained ad=nd recorded the accuracies. I have not shown the different hyperparameter models in different functions since I only changed the parameters inside the create_cnn_model and trained it each time.

Experimental Results:
Firstly(model1), the number of filters in first convolution layer in both CNN modules is set to 64, the test accuracy for the model is 0.7396. In Model2, the number of filters in first convolution layer in both CNN modules is set to 128, the test accuracy for the model is 0.7601. Another model(model 3) tried is dropout set to 0.5 instead 0.25 with 32 filters in the first conv layer. Its test accuracy is 0.6271.

Discussion: Model1 does not give that much different accuracy from original model in Question 2. However since filters in the first layer is doubled, it takes more time to compile. Model 2 gives accuracy of 0.7601. Which is slightly better than both models however, it takes a lot more time to compile. Model 3 with more dropout rate gives accuracy of 0.6271, which is worse than the dropout with 0.25.

4. The image under scope is given in Figure 4. Different outputs of different layers are visualized and examined.

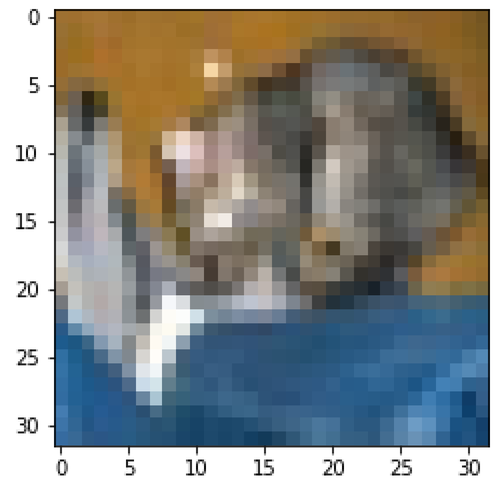


Figure 4: Test[0] image

Experimental Results:
The outputs of of different layers are visualized in the figures below.

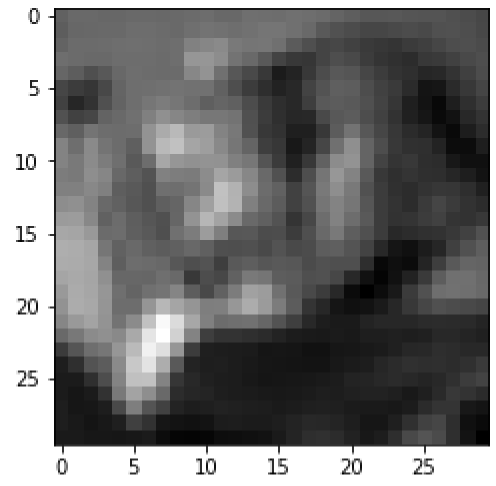


Figure 5: The output of first conv layer at left CNN branch, 1st filter.

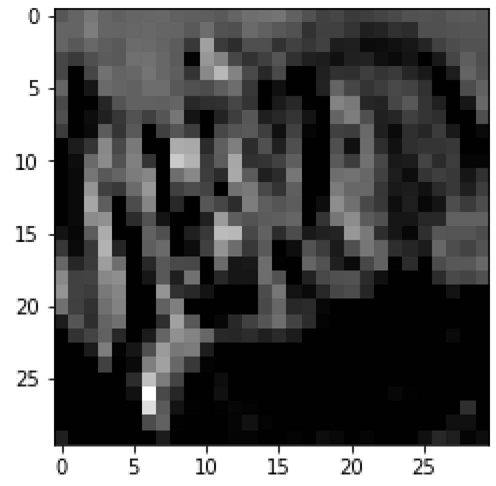


Figure 6: The output of first conv layer at left CNN branch, 3rd filter.

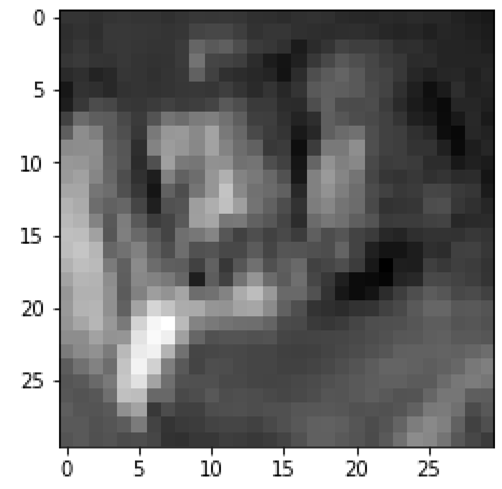


Figure 7: The output of first conv layer at right CNN branch, 1st filter.

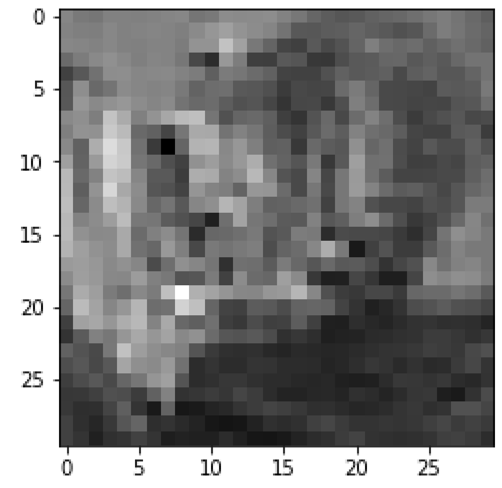


Figure 8: The output of first conv layer at right CNN branch, 10th filter.

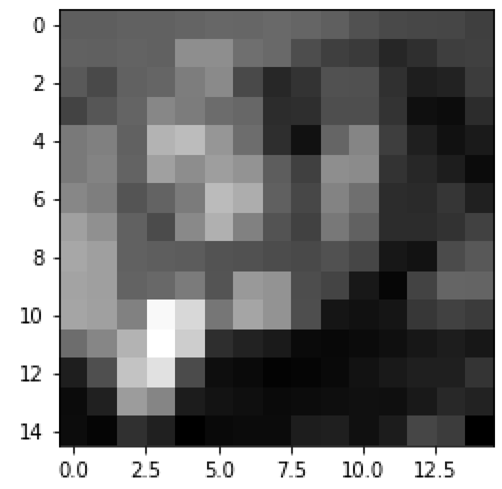


Figure 9: The output of first maxpooling layer at left CNN branch, 1st filter.

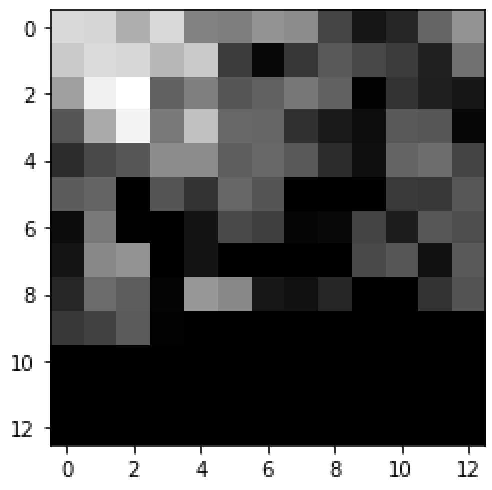


Figure 10: The output of second conv layer at left CNN branch, 1st filter.

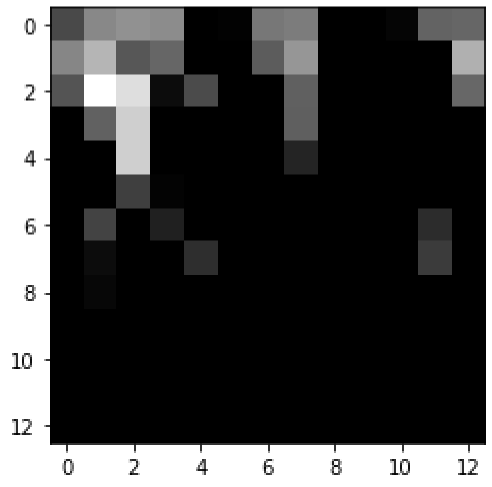


Figure 11: The output of second conv layer at left CNN branch, 5th filter.

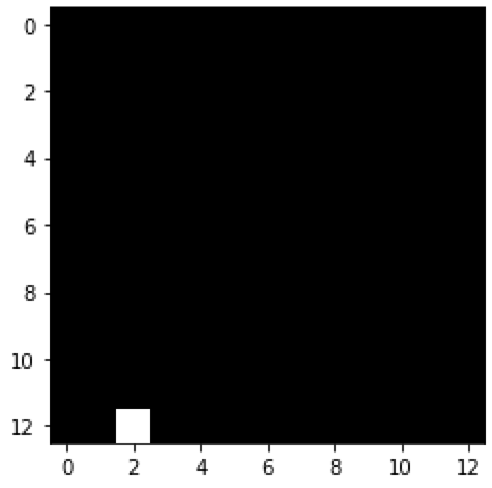


Figure 12: The output of second conv layer at right CNN branch, 1st filter.

Deniz Aytemiz
denizaytemiz@vt.edu

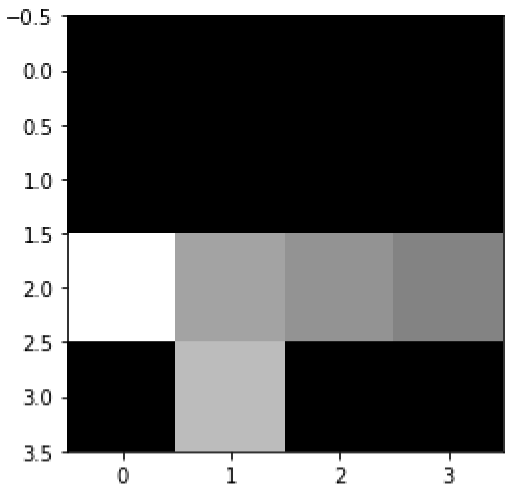


Figure 13: The output of third conv layer at left CNN branch, 1st filter.

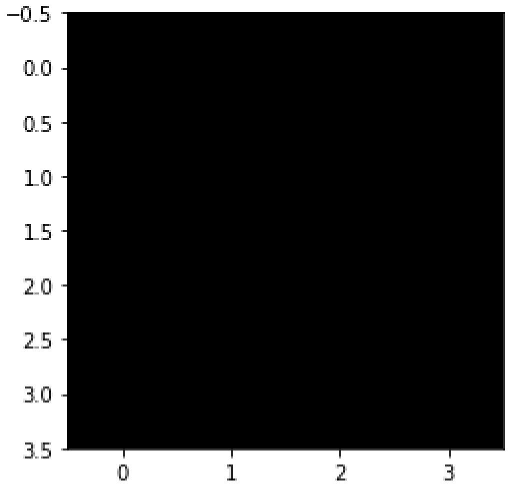


Figure 14: The output of third conv layer at right CNN branch, 1st filter.

Discussion: Figure 5 is output of first convolutional layer of left branch, it is the first filter among 32 filters. It can be observed that the original image in Figure 4 and Figure 5 are consistent with lighter colored parts are lighter and edges are more black. One can detect it belongs to test[0] image. So it's features are highlihted. Figure 7 is the first filters output of first convolutional layer in right CNN branch. One can observe that the outputs with left and right are not exactly the same but are similar. Figure 6 is the output of another filter in the first layer convolution of left branch and the feature map againis consistent with the test[0] image however it is apparent that different kernel filter is applied since contours are plotted differently. As it can be observed as dropout and especially max pooling is applied to outputs, the dimension of output reduces. And the images have more generalized look where the details are gone and it seems more blurred with indication of sharper values. In the last layers of convolution only few pixels remain in each filter, however the number of filters increase with each convolution layer and 256 filters with dimensions(2,2),(reduced to (2,2) from (4,4) after maxpooling) is fed to fully connected layer.

Part 2: This part discusses experimental results and discussions for cifar100 dataset.

2. In this part of the assignment, the 2 CNN layers are build and the output of these CNN modules are fed into a fully connected layer that outputs the classification of the image fed to the system.
The first model evaluated has the form depicted in Figure 15.

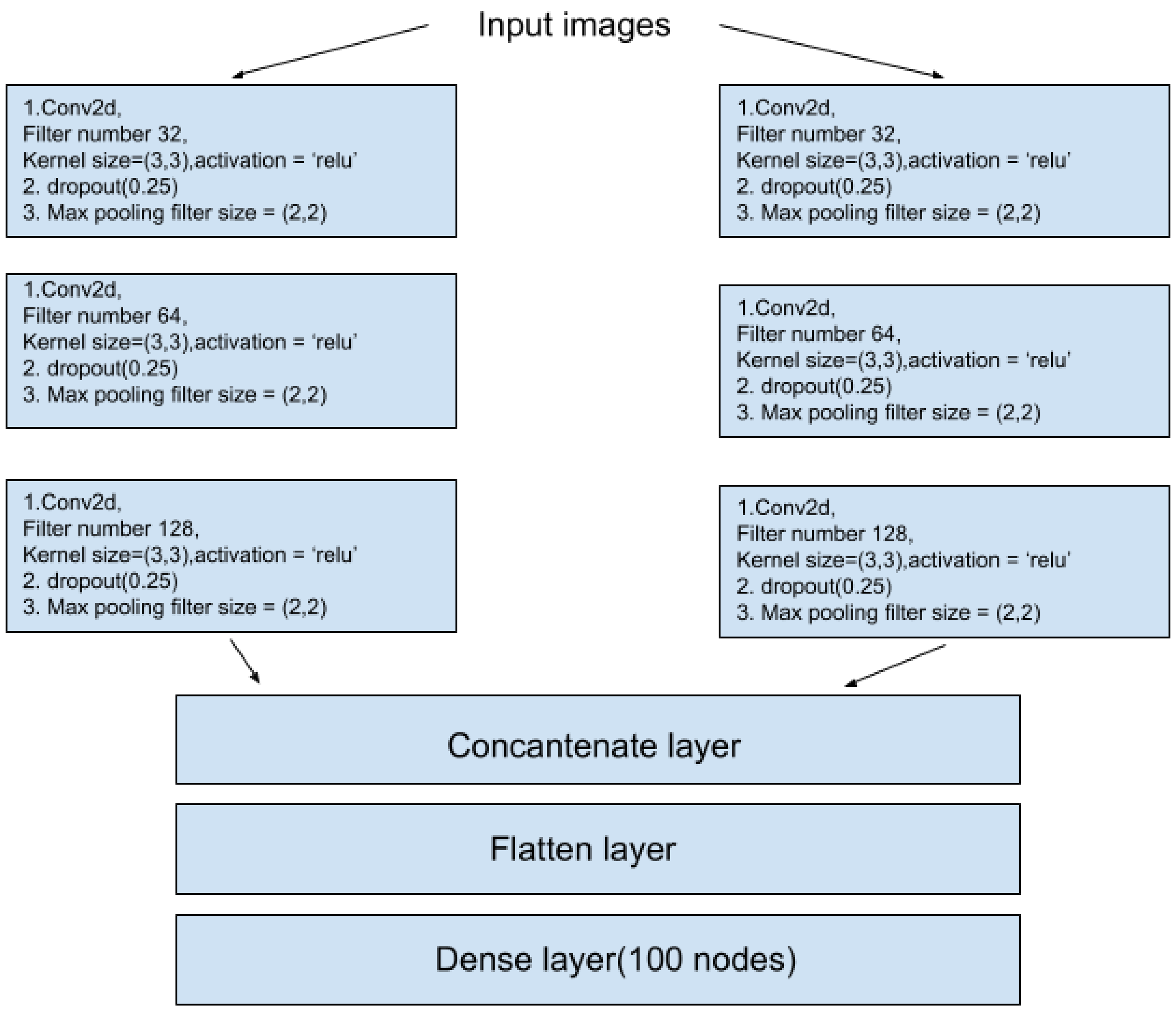

Input images
1.Conv2d,
Filter number 32,
Kernel size=(3,3),activation = 'relu'
2. dropout(0.25)
3. Max pooling filter size = (2,2)
1.Conv2d,
Filter number 32,
Kernel size=(3,3),activation = 'relu'
2. dropout(0.25)
3. Max pooling filter size = (2,2)
1.Conv2d,
Filter number 64,
Kernel size=(3,3),activation = 'relu'
2. dropout(0.25)
3. Max pooling filter size = (2,2)
1.Conv2d,
Filter number 64,
Kernel size=(3,3),activation = 'relu'
2. dropout(0.25)
3. Max pooling filter size = (2,2)
1.Conv2d,
Filter number 128,
Kernel size=(3,3),activation = 'relu'
2. dropout(0.25)
3. Max pooling filter size = (2,2)
1.Conv2d,
Filter number 128,
Kernel size=(3,3),activation = 'relu'
2. dropout(0.25)
3. Max pooling filter size = (2,2)
Concantenate layer
Flatten layer
Dense layer(100 nodes)

Figure 15: Y-network for cifar100 dataset

This model is trained as, `model.fit(trainX, trainy, batch_size=100, epochs=10, validation_data=(testX, testy))`

And the training and test accuries in each epoch are given in Figure 16.

Deniz Aytemiz
denizaytemiz@vt.edu

Experimental Results:

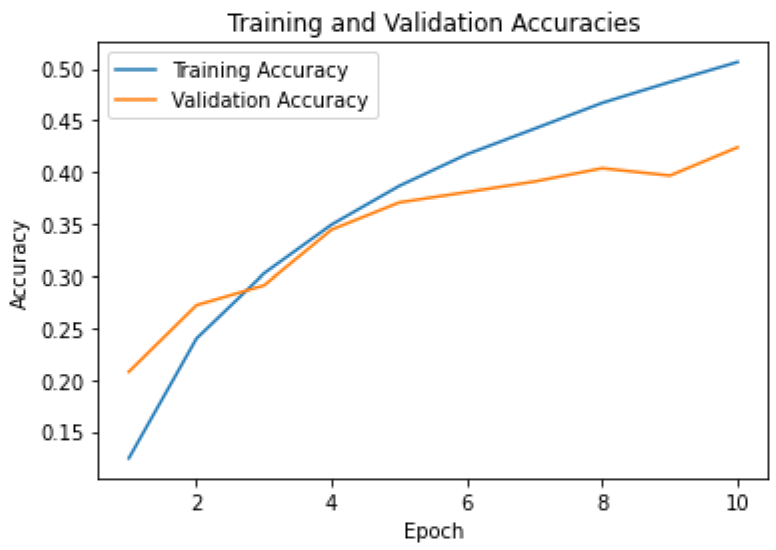
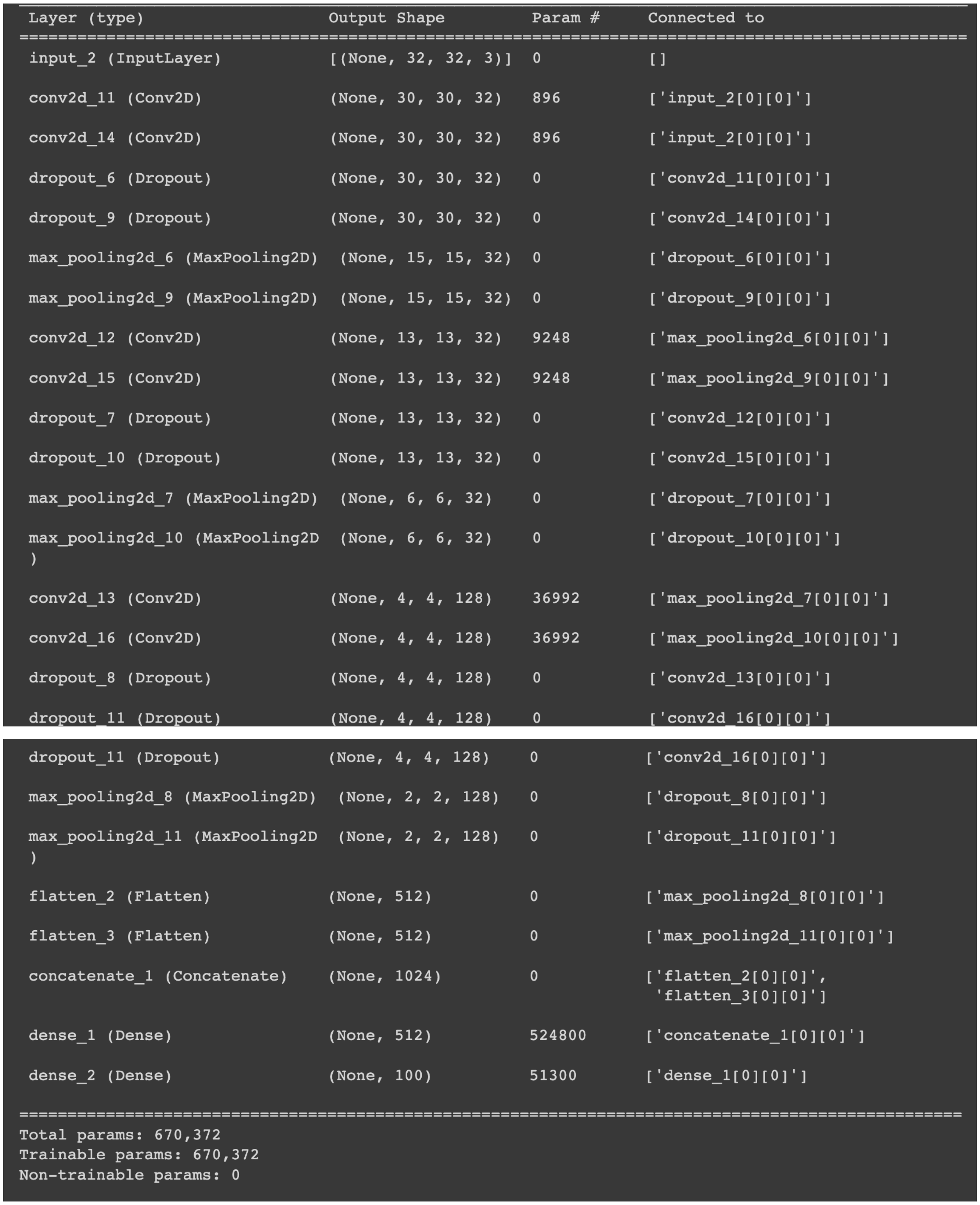


Figure 16: Test and train accuracies for model in figure 15

Discussion: The overall test accuracy for this model is 0.42399. No overfitting is observed.

3. In this part different models are examined and evaluated. First model examined has 1 more dense layer. It can be observed in Figure 17 below.

Deniz Aytemiz
denizaytemiz@vt.edu



```
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_2 (InputLayer)           [(None, 32, 32, 3)]  0           []

 conv2d_11 (Conv2D)             (None, 30, 30, 32)   896         ['input_2[0][0]']

 conv2d_14 (Conv2D)             (None, 30, 30, 32)   896         ['input_2[0][0]']

 dropout_6 (Dropout)            (None, 30, 30, 32)   0           ['conv2d_11[0][0]']

 dropout_9 (Dropout)            (None, 30, 30, 32)   0           ['conv2d_14[0][0]']

 max_pooling2d_6 (MaxPooling2D)  (None, 15, 15, 32)  0           ['dropout_6[0][0]']

 max_pooling2d_9 (MaxPooling2D)  (None, 15, 15, 32)  0           ['dropout_9[0][0]']

 conv2d_12 (Conv2D)             (None, 13, 13, 32)   9248        ['max_pooling2d_6[0][0]']

 conv2d_15 (Conv2D)             (None, 13, 13, 32)   9248        ['max_pooling2d_9[0][0]']

 dropout_7 (Dropout)            (None, 13, 13, 32)   0           ['conv2d_12[0][0]']

 dropout_10 (Dropout)           (None, 13, 13, 32)   0           ['conv2d_15[0][0]']

 max_pooling2d_7 (MaxPooling2D)  (None, 6, 6, 32)    0           ['dropout_7[0][0]']

 max_pooling2d_10 (MaxPooling2D  (None, 6, 6, 32)    0           ['dropout_10[0][0]']
 )

 conv2d_13 (Conv2D)             (None, 4, 4, 128)    36992       ['max_pooling2d_7[0][0]']

 conv2d_16 (Conv2D)             (None, 4, 4, 128)    36992       ['max_pooling2d_10[0][0]']

 dropout_8 (Dropout)            (None, 4, 4, 128)    0           ['conv2d_13[0][0]']

 dropout_11 (Dropout)           (None, 4, 4, 128)    0           ['conv2d_16[0][0]']
```

```
 dropout_11 (Dropout)           (None, 4, 4, 128)    0           ['conv2d_16[0][0]']

 max_pooling2d_8 (MaxPooling2D)  (None, 2, 2, 128)   0           ['dropout_8[0][0]']

 max_pooling2d_11 (MaxPooling2D  (None, 2, 2, 128)   0           ['dropout_11[0][0]']
 )

 flatten_2 (Flatten)            (None, 512)          0           ['max_pooling2d_8[0][0]']

 flatten_3 (Flatten)            (None, 512)          0           ['max_pooling2d_11[0][0]']

 concatenate_1 (Concatenate)    (None, 1024)         0           ['flatten_2[0][0]',
                                                                  'flatten_3[0][0]']

 dense_1 (Dense)                (None, 512)          524800      ['concatenate_1[0][0]']

 dense_2 (Dense)                (None, 100)          51300       ['dense_1[0][0]']

==================================================================================================
Total params: 670,372
Trainable params: 670,372
Non-trainable params: 0
```

Figure 17 : Model1

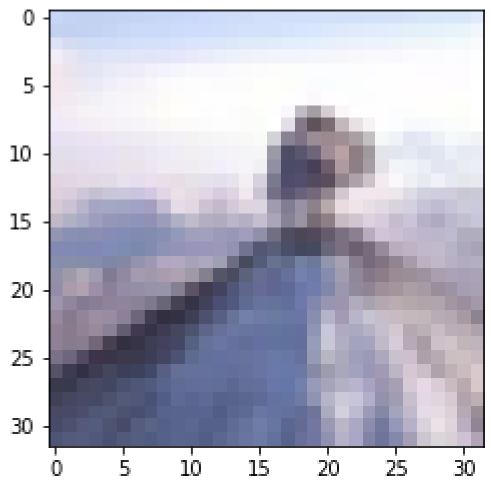
Experimental Results: Model 1 used compiler ‘adam’ instead of rmsprop and has 2 dense layers, the test accuracy Model1 gives, 0.05209. Another model used 1 dense layer and dropout rate of 0.5, which resulted in accuracy of 0.3811, its optimizer is ‘adam’.

Discussion: Comparing the 1 dense layer and 2-dense layered model, 1 dense layered model gives much better accuracy and less compile time. The optimizer adam instead of rmsprop. The accuracy it gives is 0.05209. Which is significantly lower than 1 dense layer with optimizer rmsprop model. I think it is not due to optimizer change since I had acquired good accuracies with both optimiizers in same models, but it is probably the first dense layer having 512 neurons whereas concantenate outputs (none,1024) shaped output.

Comparing the dropouts rates, dropout rate with 0.25 gives better accuracy then 0.5 which is 0.42 and 0.38 respectively.

4. In this part, the visualization of outputs of convolutional layers are given for cifar100 datasets in the figures below.

Experimental Results: The outputs are given in the below figures.



.

Figure 18: Test[0] image

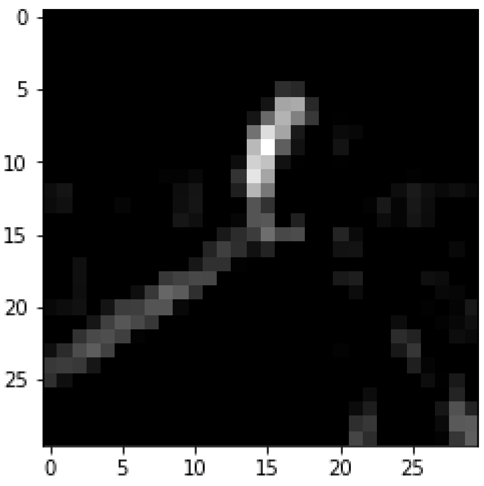


Figure 19: The output of first conv layer at left CNN branch, 1st filter.

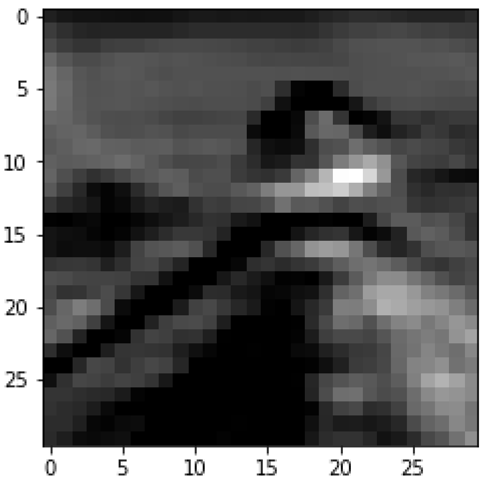


Figure 20: The output of first conv layer at left CNN branch, 3rd filter.

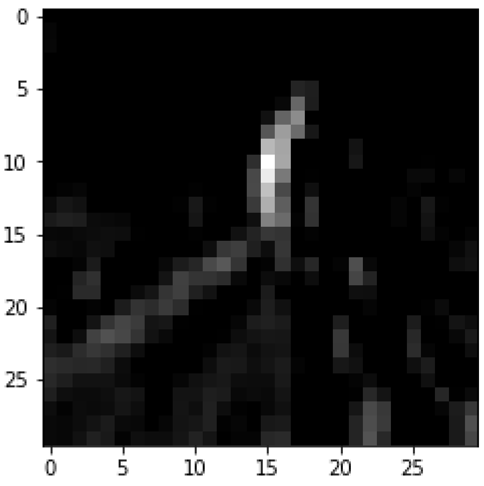


Figure 21: The output of first conv layer at right CNN branch, 1st filter.

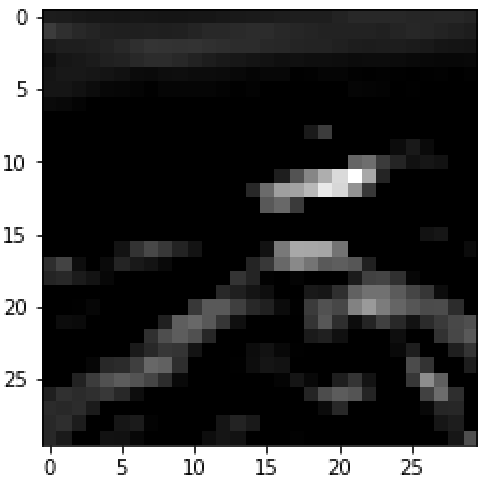


Figure 22: The output of first conv layer at right CNN branch, 10th filter.

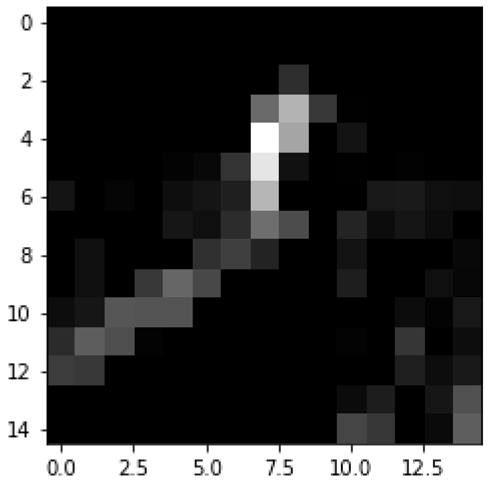


Figure 23: The output of first maxpooling layer at left CNN branch, 1st filter.

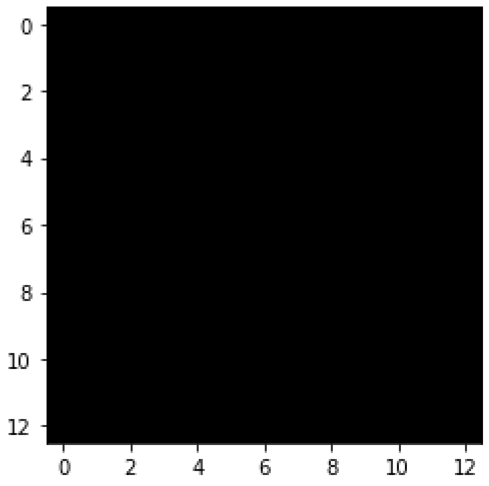


Figure 24: The output of second conv layer at left CNN branch, 1st filter.

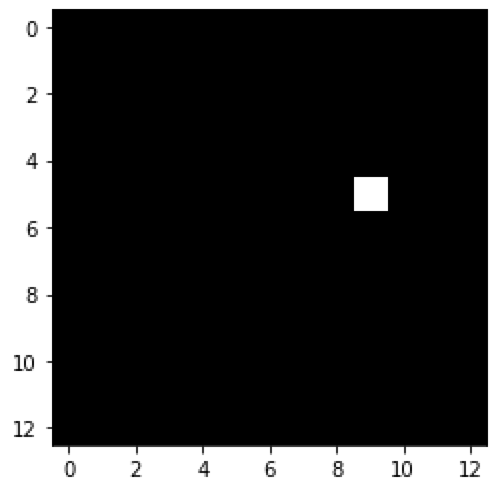


Figure 25: The output of second conv layer at left CNN branch, 5th filter.

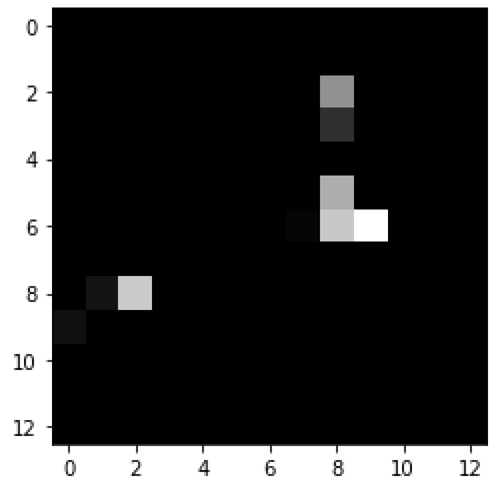


Figure 26: The output of second conv layer at right CNN branch, 1st filter.

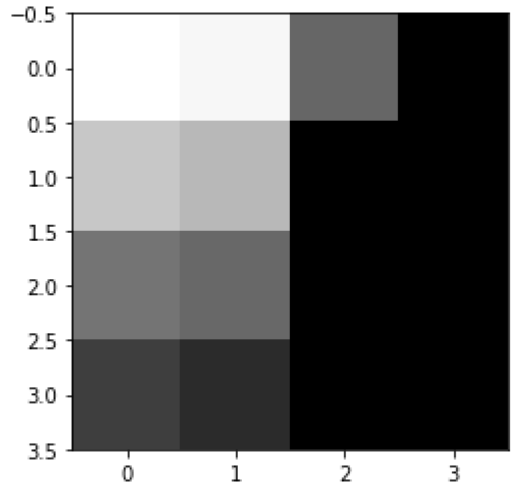


Figure 27: The output of third conv layer at left CNN branch, 1st filter.

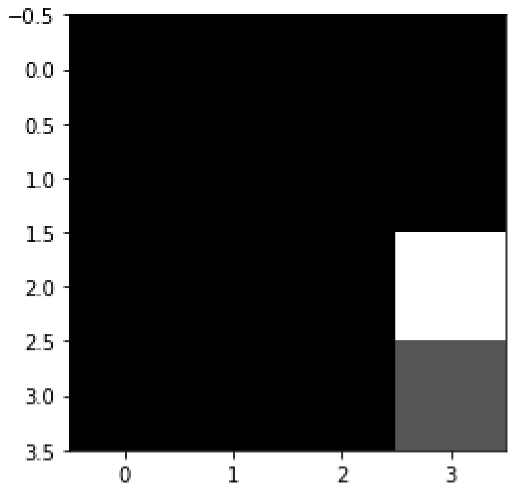


Figure 28: The output of third conv layer at right CNN branch, 1st filter.

Discussion: As in the cifar10 daytaset one can observe that output of the first layer convolution shows different characteristics of the test[0] image. And as the layers increases number of filters increase but dimenions reduced.