COMP5623 Coursework on Image Classification and Visualizations with Convolutional Neural Networks – ImageNet10

Name	**Praveen Gopal Reddy**

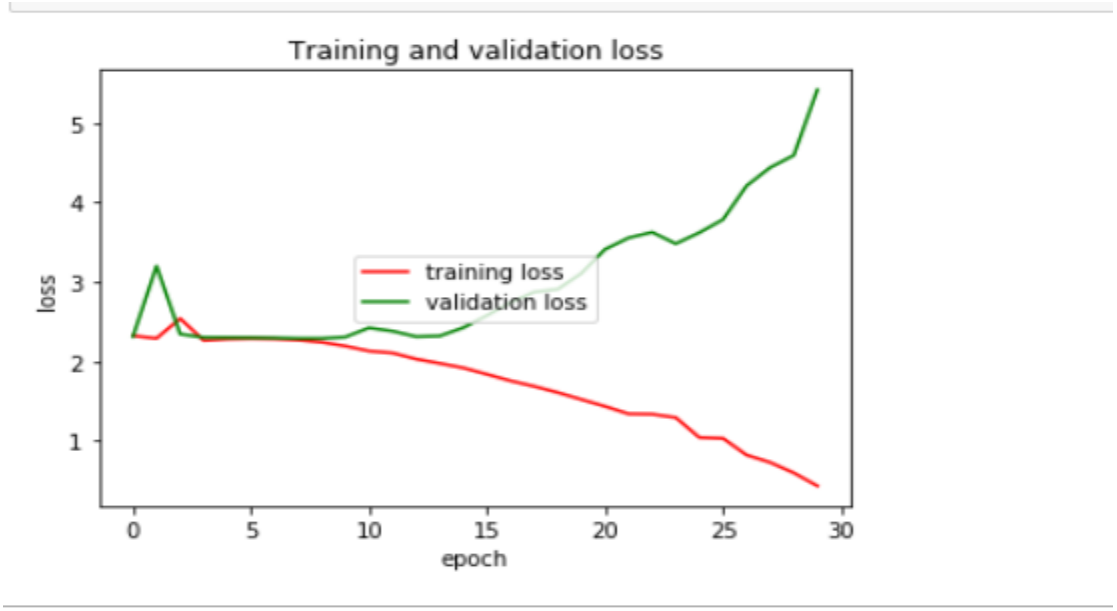
QUESTION I [55 marks]

1.1 Single-batch training [16 marks]

1.1.1. Display graph 1.1.1 (training & validation loss over training epochs) and briefly explain what is happening and why. [4 marks]

From the graph we can see that training loss is decreasing and and almost touching zero from epoch 7 to 29 which means our model learns patterns extremely good in training set. Whereas validation loss keeps increasing right from epoch 1 which means model is performing bad in predictions to new unseen data present in validation set. Additionally, our model is not generalizing well enough on the validation set.

1.1.2 Display graph 1.1.2 (training & validation loss over training epochs, with modified architecture) and explain how and why it shows that the model is overfitting the training batch. [8 marks]




```
epoch: 0 training loss:  2.321  validation loss:  2.309 Accuracy:  9.3%
epoch: 1 training loss:  2.287  validation loss:  3.198 Accuracy:  9.9%
epoch: 2 training loss:  2.538  validation loss:  2.341 Accuracy:  9.4%
epoch: 3 training loss:  2.261  validation loss:  2.300 Accuracy:  11.6%
epoch: 4 training loss:  2.278  validation loss:  2.298 Accuracy:  14.4%
epoch: 5 training loss:  2.284  validation loss:  2.296 Accuracy:  15.1%
epoch: 6 training loss:  2.280  validation loss:  2.293 Accuracy:  15.7%
epoch: 7 training loss:  2.266  validation loss:  2.287 Accuracy:  16.7%
epoch: 8 training loss:  2.239  validation loss:  2.287 Accuracy:  14.6%
epoch: 9 training loss:  2.190  validation loss:  2.305 Accuracy:  13.6%
epoch: 10 training loss:  2.128  validation loss:  2.420 Accuracy:  14.3%
epoch: 11 training loss:  2.104  validation loss:  2.378 Accuracy:  17.3%
epoch: 12 training loss:  2.027  validation loss:  2.310 Accuracy:  19.1%
epoch: 13 training loss:  1.973  validation loss:  2.318 Accuracy:  19.7%
epoch: 14 training loss:  1.916  validation loss:  2.422 Accuracy:  20.6%
epoch: 15 training loss:  1.836  validation loss:  2.575 Accuracy:  20.2%
epoch: 16 training loss:  1.753  validation loss:  2.738 Accuracy:  20.2%
epoch: 17 training loss:  1.685  validation loss:  2.871 Accuracy:  20.8%
epoch: 18 training loss:  1.606  validation loss:  2.906 Accuracy:  20.7%
epoch: 19 training loss:  1.519  validation loss:  3.105 Accuracy:  20.7%
epoch: 20 training loss:  1.433  validation loss:  3.407 Accuracy:  21.8%
epoch: 21 training loss:  1.338  validation loss:  3.552 Accuracy:  21.2%
epoch: 22 training loss:  1.333  validation loss:  3.622 Accuracy:  20.9%
epoch: 23 training loss:  1.292  validation loss:  3.478 Accuracy:  23.1%
epoch: 24 training loss:  1.040  validation loss:  3.618 Accuracy:  22.6%
epoch: 25 training loss:  1.031  validation loss:  3.782 Accuracy:  23.8%
epoch: 26 training loss:  0.820  validation loss:  4.207 Accuracy:  22.6%
epoch: 27 training loss:  0.727  validation loss:  4.437 Accuracy:  21.8%
epoch: 28 training loss:  0.596  validation loss:  4.590 Accuracy:  23.3%
epoch: 29 training loss:  0.432  validation loss:  5.415 Accuracy:  22.7%
```

from graph we can see that there is not much gap difference between valid loss and train loss from epoch 2 to epoch 8 as they are almost converging and at this point, we cannot tell it is overfitting or not. As epochs are processing, the train loss is decreasing which means it is doing good in learning parameters, while validation loss gradually increasing step by step and interestingly accuracy making some variations but eventually it will tend to decrease as valid loss increasing and epochs increasing.

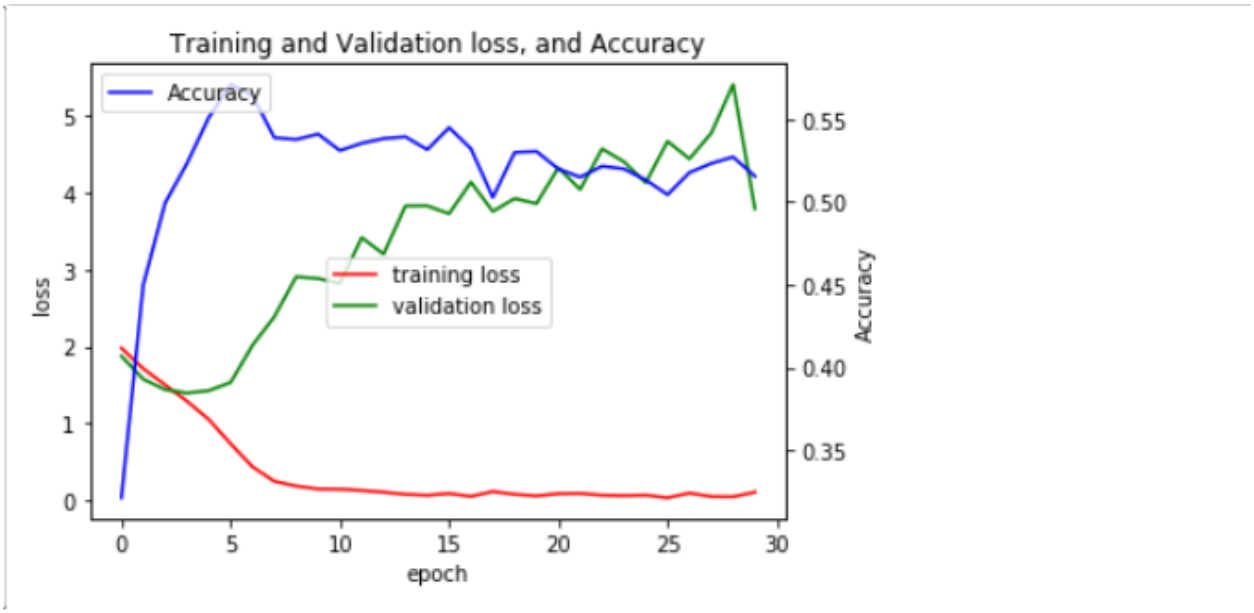
Finally, model will start to overfit from epoch 26 as train loss nearing to zero and validation loss surging and validation accuracy decreasing where our model doing poor generalization on validation set. The reasons could be our model is too complex, data has noise or size of data we use for training not enough or we did not train long enough.

1.1.3 Fill in table 1.1.3 (your adjusted architecture after single-batch training), adding rows and columns, as necessary. [4 marks]

Input channels	Output channels	Layer type	Kernel size	stride
3	32	Conv2d	5	
	32	ReLU		
3	32	Maxpool	2	2
32	64	Conv2d	3	
	64	ReLU		
32	64	Maxpool	2	2
64	128	Conv2d	3	
	128	ReLU		
64	128	Maxpool	2	2
128	256	Conv2d	3	
	256	ReLU		
128	256	Maxpool	2	2
9216	4000	Linear		
	4000	ReLU		
4000	900	Linear		
	900	ReLU		
900	90	Linear		
	90	ReLU		
90	10	Linear		
	10	ReLU		

1.2 Fine-tuning on full dataset [18 marks]

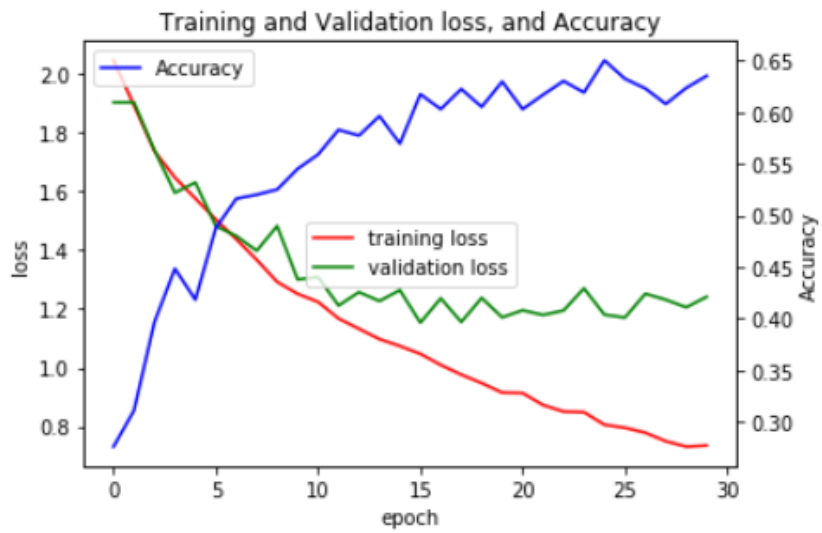
1.2.1 Display graph 1.2.1 and indicate what the optimal number of training epochs is and why. [4 marks]




```
epoch: 0 training loss:  1.979  validation loss:  1.874  Accuracy:  32.1%
epoch: 1 training loss:  1.711  validation loss:  1.574  Accuracy:  45.0%
epoch: 2 training loss:  1.500  validation loss:  1.438  Accuracy:  49.9%
epoch: 3 training loss:  1.289  validation loss:  1.395  Accuracy:  52.3%
epoch: 4 training loss:  1.050  validation loss:  1.426  Accuracy:  55.1%
epoch: 5 training loss:  0.734  validation loss:  1.532  Accuracy:  57.1%
epoch: 6 training loss:  0.435  validation loss:  2.016  Accuracy:  56.4%
epoch: 7 training loss:  0.246  validation loss:  2.387  Accuracy:  53.9%
epoch: 8 training loss:  0.184  validation loss:  2.907  Accuracy:  53.8%
epoch: 9 training loss:  0.146  validation loss:  2.884  Accuracy:  54.1%
epoch: 10 training loss:  0.144  validation loss:  2.820  Accuracy:  53.1%
epoch: 11 training loss:  0.127  validation loss:  3.416  Accuracy:  53.6%
epoch: 12 training loss:  0.105  validation loss:  3.202  Accuracy:  53.8%
epoch: 13 training loss:  0.077  validation loss:  3.828  Accuracy:  53.9%
epoch: 14 training loss:  0.064  validation loss:  3.829  Accuracy:  53.2%
epoch: 15 training loss:  0.087  validation loss:  3.727  Accuracy:  54.5%
epoch: 16 training loss:  0.050  validation loss:  4.137  Accuracy:  53.2%
epoch: 17 training loss:  0.113  validation loss:  3.757  Accuracy:  50.3%
epoch: 18 training loss:  0.077  validation loss:  3.924  Accuracy:  53.0%
epoch: 19 training loss:  0.057  validation loss:  3.859  Accuracy:  53.1%
epoch: 20 training loss:  0.085  validation loss:  4.324  Accuracy:  52.0%
epoch: 21 training loss:  0.089  validation loss:  4.042  Accuracy:  51.5%
epoch: 22 training loss:  0.066  validation loss:  4.570  Accuracy:  52.2%
epoch: 23 training loss:  0.060  validation loss:  4.404  Accuracy:  52.0%
epoch: 24 training loss:  0.066  validation loss:  4.129  Accuracy:  51.3%
epoch: 25 training loss:  0.032  validation loss:  4.669  Accuracy:  50.4%
epoch: 26 training loss:  0.093  validation loss:  4.440  Accuracy:  51.8%
epoch: 27 training loss:  0.048  validation loss:  4.775  Accuracy:  52.3%
epoch: 28 training loss:  0.045  validation loss:  5.409  Accuracy:  52.7%
epoch: 29 training loss:  0.105  validation loss:  3.792  Accuracy:  51.6%
```

The optimal number of epochs would be 10 because from graph we can see that both train loss and valid loss started well with less difference gap and accuracy increasing till epoch 5 but as train loss decreasing, valid loss increasing from epoch 6 and continues to increase its loss and decrease its accuracy. As the number of epochs increases beyond 11, training set loss decreases and becomes nearly zero. Whereas validation loss increases depicting the overfitting of the model on training data. So, our optimal epochs are 10.

1.2.2 Describe in detail your fine-tuning process on the complete dataset, including any adjustments you made to the network or training process to increase prediction accuracy. Explain why these adjustments increased accuracy. [10 marks]




```
epoch: 0 training loss:  2.046  validation loss:  1.904  Accuracy:  27.6%
epoch: 1 training loss:  1.893  validation loss:  1.904  Accuracy:  31.1%
epoch: 2 training loss:  1.737  validation loss:  1.740  Accuracy:  39.7%
epoch: 3 training loss:  1.647  validation loss:  1.596  Accuracy:  44.8%
epoch: 4 training loss:  1.576  validation loss:  1.631  Accuracy:  41.8%
epoch: 5 training loss:  1.505  validation loss:  1.482  Accuracy:  48.8%
epoch: 6 training loss:  1.438  validation loss:  1.449  Accuracy:  51.6%
epoch: 7 training loss:  1.367  validation loss:  1.398  Accuracy:  52.0%
epoch: 8 training loss:  1.292  validation loss:  1.483  Accuracy:  52.5%
epoch: 9 training loss:  1.252  validation loss:  1.300  Accuracy:  54.5%
epoch: 10 training loss:  1.224  validation loss:  1.309  Accuracy:  55.9%
epoch: 11 training loss:  1.168  validation loss:  1.212  Accuracy:  58.3%
epoch: 12 training loss:  1.133  validation loss:  1.257  Accuracy:  57.7%
epoch: 13 training loss:  1.098  validation loss:  1.227  Accuracy:  59.6%
epoch: 14 training loss:  1.074  validation loss:  1.264  Accuracy:  56.9%
epoch: 15 training loss:  1.047  validation loss:  1.153  Accuracy:  61.7%
epoch: 16 training loss:  1.009  validation loss:  1.237  Accuracy:  60.3%
epoch: 17 training loss:  0.977  validation loss:  1.156  Accuracy:  62.2%
epoch: 18 training loss:  0.948  validation loss:  1.238  Accuracy:  60.5%
epoch: 19 training loss:  0.915  validation loss:  1.172  Accuracy:  62.9%
epoch: 20 training loss:  0.914  validation loss:  1.196  Accuracy:  60.3%
epoch: 21 training loss:  0.873  validation loss:  1.180  Accuracy:  61.7%
epoch: 22 training loss:  0.851  validation loss:  1.195  Accuracy:  63.0%
epoch: 23 training loss:  0.849  validation loss:  1.270  Accuracy:  61.9%
epoch: 24 training loss:  0.806  validation loss:  1.181  Accuracy:  65.0%
epoch: 25 training loss:  0.796  validation loss:  1.171  Accuracy:  63.2%
epoch: 26 training loss:  0.779  validation loss:  1.252  Accuracy:  62.3%
epoch: 27 training loss:  0.750  validation loss:  1.232  Accuracy:  60.8%
epoch: 28 training loss:  0.732  validation loss:  1.206  Accuracy:  62.3%
epoch: 29 training loss:  0.735  validation loss:  1.242  Accuracy:  63.5%
```

As you can see the validation accuracy of graph increased from 51% to 63% compared to graph in 1.21. I think if we increase epochs number to 60-70,we can achieve accuracy more than 80%

The fine-tuning process that I made on dataset and network using data augmentation and regularization techniques are:

a. Added random affine transformation.
b. Added random horizontal flip flop transformation.
c. Added dropout on network.

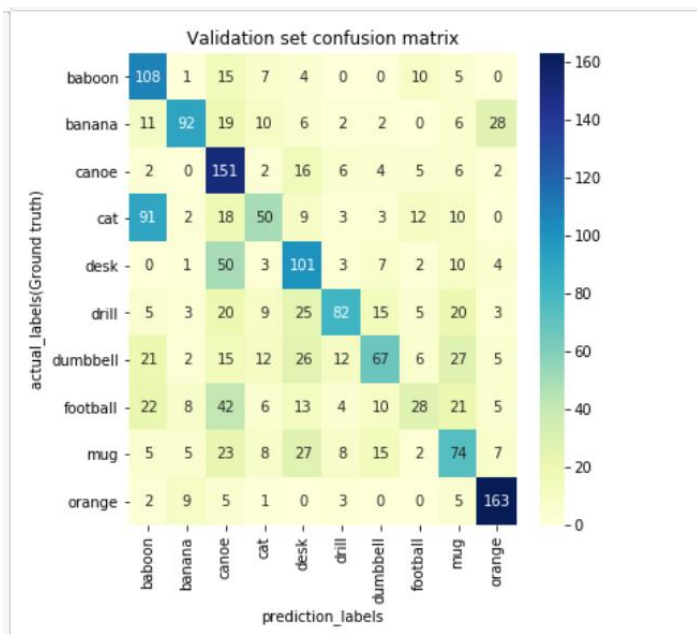
The parameter that I used for affine transformation are rotation, scaling and translation, by doing this it preserves lines, planes, and points on affine space. The idea is to reduce minimizing loss and reduce overfitting and hence increase accuracy.

Random flip will flip the images horizontally with a probability of 0.2 so that our model will be exposed to new variations of image during training which will increase accuracy.

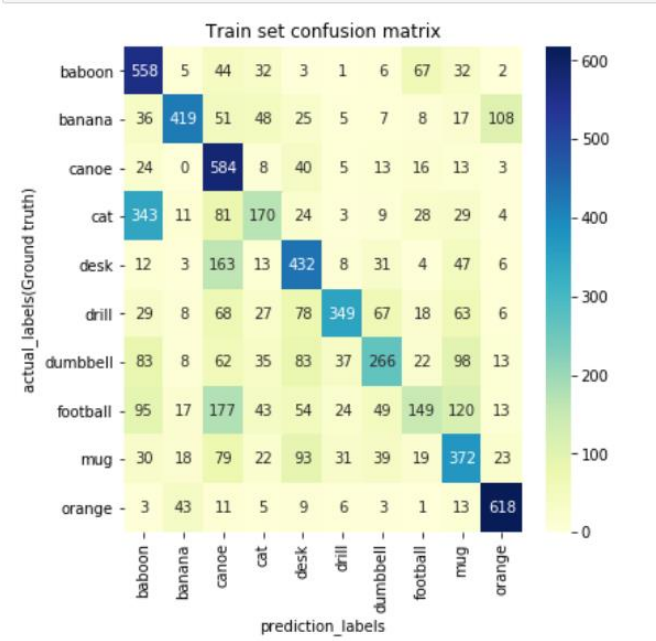
I have used dropout three times in model network with probability of 0.2 and 0.3. dropout will remove neurons from hidden network and reduces computations which makes network simpler and thus reduce overfitting of the training module.

1.2.3 Display two confusion matrices 1.2.3 (one each for complete validation set and complete training set) for your final trained model and interpret what is shown. [4 marks]

Validation set confusion matrix.



Train set confusion matrix.



From validation set confusion matrix most of the boxes (lower triangle and upper triangle) are colored in yellow where values range from 0 to 20. The worst prediction is, where Cat is identified as baboon for 91 times. The highest correct prediction is orange where 163 images are correctly predicted as oranges. Second is canoe and third is baboon.

The diagonal here is used to identify correct predictions. The higher the diagonal values in confusion matrix the better, indicating many correct predictions.

From train set confusion matrix, badly predicted labels against actual labels ranging from values 0 to 340, is colored in yellow and green. The highest correct prediction is orange where 618 images are correctly predicted as oranges. Second is baboon and third is canoe. The worst prediction is, where cat is identified as baboon for 343 times.

1.3 Evaluation and code [21 marks]

1.3.1 Please include *[my_student_username]_test_preds.csv* with your final submission. [8 marks]

1.3.2 Please submit all relevant code you wrote for Question I in Python file *[my_student_username]_q1.py*. No need to include the config or ImageNet10 files. [13 marks]

No response needed here.

QUESTION II [45 marks]

2.1 Preparing the pre-trained network [20 marks]

2.1.1 Read through the provided template code for the AlexNet model *alexnet.py*. What exactly is being loaded in line 59? [2 marks]

Loads the Alexnet model and downloads pre-trained weights using pytorch hub model repository if not already downloaded. It downloads the model from pytorch version 0.6

2.1.2 Write the code in *explore.py* after line 50 to read in the image specified in the variable `args.image_path` and pass it through a single forward pass of the pre-trained AlexNet model. [5 marks]

2.1.3 Fill in function `extract_filter()` after line 84 extracting the filters from a given layer of the pre-trained AlexNet. [4 marks]

2.1.4 Fill in function `extract_feature_maps()` after line 105 extracting the feature maps from the convolutional layers of the pre-trained AlexNet. [6 marks]

Please submit all your Question II code in a Python file *[my_student_username]_explore.py*.

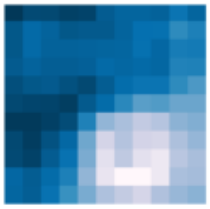
No response needed here.

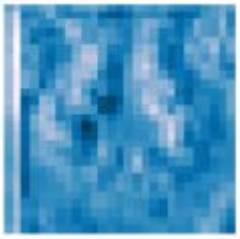
2.1.5 Describe in words, not code, how you ensure that your filters and feature maps are pairs; that the feature maps you extract correspond to the given filter. [3 marks]

2.2 Visualizations [25 marks]

2.2.1 For three input images of different classes, show three pairs of filters and corresponding feature maps, each from a different layer in AlexNet. Indicate which layers you chose. For each pair, briefly explain what the filter is doing (for example: horizontal edge detection) which should be confirmed by the corresponding feature map. [15 marks]

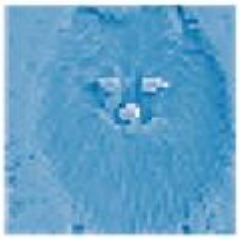
Image #1, class: __Tiger_____

	Filter	Feature map	Brief explanation
Early layer	Filter1: R channel  Filter2: R channel 	Conv2d feature map:   Relu feature map after activation:  	Layer 0: for Filters Layer 0,1: for feature maps The filter1 and filter 2 has light and dark edges. so, most of the information present in the image is retained. the first layers usually act as edge detectors.
Intermediate layer	Filter pairs	Conv2d feature map:	Layer 3: for Filters Layer 3,4: for feature maps

	 	  Relu feature map  	Here filters have lots of colors detecting entire image features
Deep layer	Filter pairs 	Conv2d feature map: 	Layer 10: for Filters Layer 10,11: for feature maps Here filters are detecting edges

		 Relu feature  	

Image #2, class: __ Pomeranian __

	Filter	Feature map	Brief explanation
Early layer	Filter pairs  	Conv2d feature map:   Relu feature map	Layer 0: for Filters Layer 0,1: for feature maps Here filters are detecting prominent features in an image like eyes and nose

		 	
Intermediate layer	 	Conv2d feature map:   Relu feature map  	Layer 6: for Filters Layer 6,7: for feature maps Here filters are detecting corners and face features in an image like eyes and nose
Deep layer	Filter pairs	Con2d feature maps	Layer 10: for Filters Layer 10,11: for feature maps

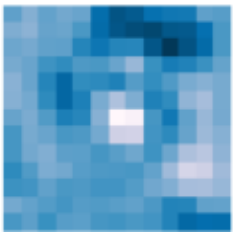
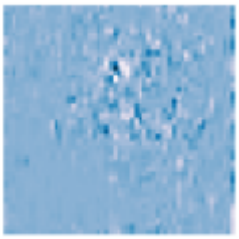
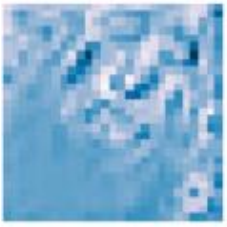
	 	  Relu feature map  	Here filters are detecting corners and edges in an image

Image #3, class: _chimpanzee_____

	Filter	Feature map	Brief explanation
Early layer	Filter pairs	Conv2d feature map:	Layer 0: for Filters Layer 0,1: for feature maps

	 	  Relu feature map 	Here filters are detecting horizontal. And vertical edges
Intermediate layer	Filter pairs  	Conv2d feature map:  	Layer 3: for Filters Layer 3,4: for feature maps Here filters have lots of colors detecting entire image features.

		Relu feature map 	
Deep layer	Filter pairs  	Conv2d feature map:   Relu feature map: 	Layer 10: for Filters Layer 10,11: for feature maps Here filters are detecting corners in image.

2.2.2 Comment on how the filters and feature maps change with depth into the network. [5 marks]

At early layers number of filters is less and kernel size is big, so these filters try to capture edges, lines, and curves. At intermediate layers kernel size decrease and filters, feature maps will

increase. So, filters capture important features like nose, eyes, ears or even retain an image. The feature maps become sparser as we go deeper, meaning the filters detect less features. In the first layers it detects simple shapes, and every image contains those. But as we go deeper, we start looking for more complex stuff like dog tail and they do not appear in every image.

Marks reserved for overall quality of report. [5 marks]

No response needed here.