Assignment 3 Report

This is an outline for your report to ease the amount of work required to create your report. Jupyter notebook supports markdown, and I recommend you to check out this cheat sheet (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet). If you are not familiar with markdown.

Before delivery, **remember to convert this file to PDF**. You can do it in two ways:

1. Print the webpage (ctrl+P or cmd+P)
2. Export with latex. This is somewhat more difficult, but you'll get somehwat of a "prettier" PDF. Go to File -> Download as -> PDF via LaTeX. You might have to install nbconvert and pandoc through conda; `conda install nbconvert pandoc`.

For TA: The models were trained using the enclosed ipynb: `TDT4265_Assignment3_Group190.ipynb`. I have copied the relevant code into .py files if you require them to run but there is no guarantee that it would work as my computer has difficulty setting up a local copy of pytorch. The ipynb would run if it is required to replicate my work. It was trained on Google Colab with a P100.

Task 1

task 1a - 1c)

(a) Padding = 1 is used, stride = 1 is used to get a 3x5 feature map.

∴ Padded image:

0	0	0	0	0	0	0
0	1	0	2	3	1	0
0	3	2	0	7	0	0
0	0	6	1	1	4	0
0	0	0	0	0	0	0

Sobel Kernel

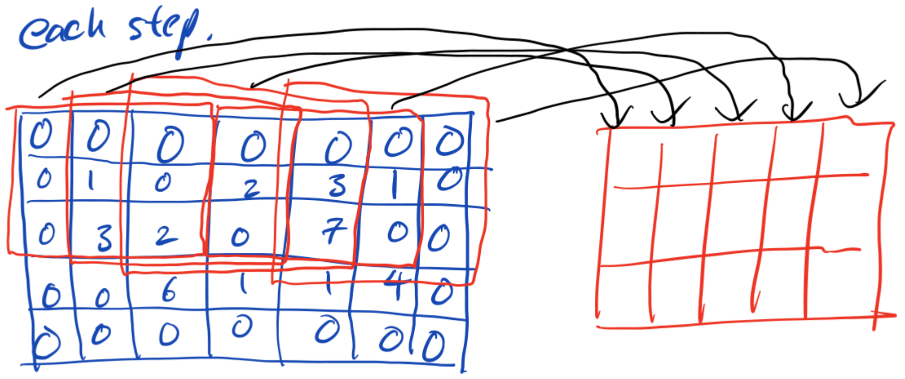
-1	0	1
-2	0	2
-1	0	1

To apply the kernel, we first need to flip up and flip left.

∴ Resultant kernel:

1	0	-1
2	0	-2
1	0	-1

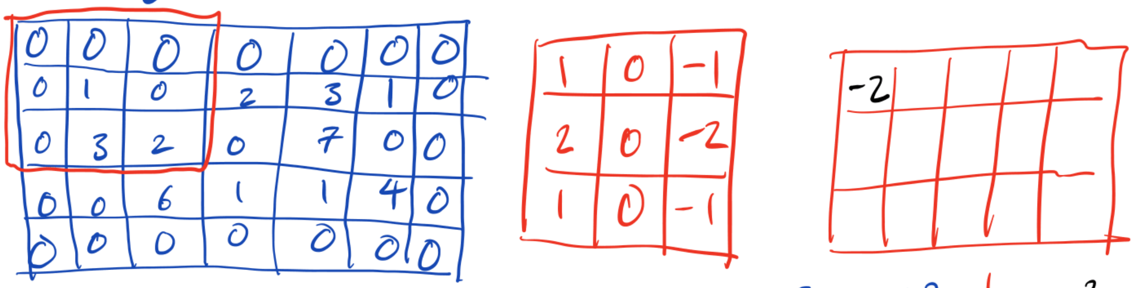
The kernel is applied as a sliding window, moving 1 square each step.



The kernel and portion of the image undergoes an element-wise multiplication and is summed to get the value of the top left cell. As the window

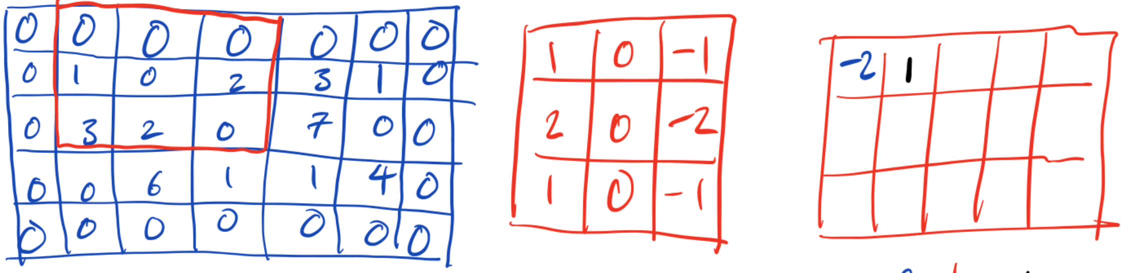
Shifts right, the resultant value of the convolution is the value of the next cell on the right. This also occurs as the window shifts down, resulting in the value to be the one belonging to the cell below.

E.g.



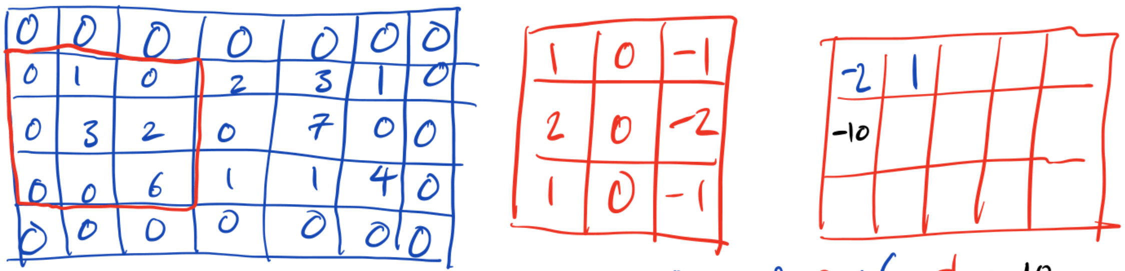
$0\times 1 + 0\times 0 + 0\times -1 + 0\times 2 + 1\times 0 + 0\times -2 + 0\times 1 + 3\times 0 + 2\times -1 = -2$

$0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -2$



$0\times 1 + 0\times 0 + 0\times -1 + 1\times 2 + 0\times 0 + 2\times -2 + 3\times 1 + 2\times 0 + 0\times -1 = 1$

$0 \quad 0 \quad 0 \quad 2 \quad 0 \quad -4 \quad 3 \quad 0 \quad 0$



$0\times 1 + 1\times 0 + 0\times -1 + 0\times 2 + 3\times 0 + 2\times -2 + 0\times 1 + 0\times 0 + 6\times -1 = -10$

$0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -4 \quad 0 \quad 0 \quad -6$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 3 & 1 & 0 \\ 0 & 3 & 2 & 0 & 7 & 0 & 0 \\ 0 & 0 & 6 & 1 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} -2 & 1 & -11 & 2 & 13 \\ -10 & 4 & -8 & -2 & 18 \\ -14 & 1 & 5 & -6 & 9 \end{bmatrix}$$

After applying the kernel across the padded this is the resultant feature map.

$$\begin{bmatrix} -2 & 1 & -11 & 2 & 13 \\ -10 & 4 & -8 & -2 & 18 \\ -14 & 1 & 5 & -6 & 9 \end{bmatrix}$$

(b) Max Pooling. It chooses the highest activation in it's kernel, reducing sensitivity to small pixel shifts.

(c) With the formula,

$$W_2 = (W_1 - F_W + 2P_W)/S_W + 1$$

and $W_2 = W_1 = 5$, $S_W = 1$, $F_W = 5$

$$W_2 - 1 = 2P_W$$
$$4 = 2P_W$$

task 1c - 1e)

$P_w = 2$

Similarly for height,

$P_h = 2$

$\therefore$ Padding of 2.

1d) Width:

$504 = (512 - F_W + 0)/1 + 1$

$503 = 512 - F_W$

$F_W = 9$

Height:

$504 = (512 - F_H + 0)/1 + 1$

$503 = 512 - F_H$

$F_H = 9$

Kernel dimension: $(9) \times (9)$

1e) $W_2 = (504 - 2 + 0)/2 + 1$

$= 252$

$H_2 = (504 - 2 + 0)/2 + 1$

$= 252$

Spatial dimension: $(252) \times (252)$

task 1f - 1g)

1f) $W_2 = (252 - 3 + 0)/1 + 1$

$= 250$

$H_2 = (252 - 3 + 0)/1 + 1$

$= 250$

Spatial Dimension: $(250) \times (250)$

1g) Filter of 5×5, padding $= 2$ and stride $= 1$

$\hookrightarrow$ Same padding, dimeasions stay same.

MaxPool2D and Flatten has no trainable parameters. MaxPool2D halves the dimension given a kernel size of 2.

Layer 1: $(5 \times 5 + 1) \times 32 = 832$ parameters.

$\text{Dimensions}_{out} = [16, 16, 32]$

Layer 2: $(5 \times 5 + 1) \times 64 = 1664$ parameters.

$\text{Dimensions}_{out} = [8, 8, 64]$

Layer 3: $(5 \times 5 + 1) \times 128 = 3328$ parameters.

$\text{Dimension}_{out} = [4, 4, 128]$

After flattening,

Dimensions = [4x4x128,] = [2048,]

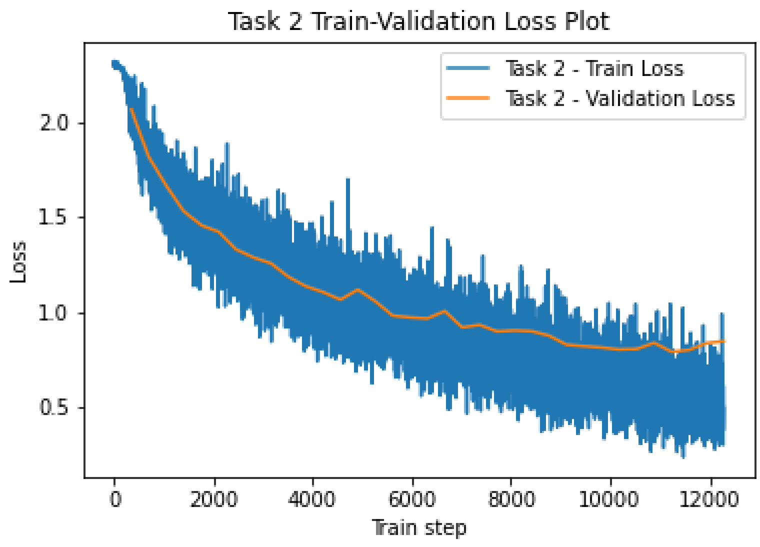
Layer 4: $(2048+1) \times 64 = 131{,}136$ params

Layer 5: $(64+1) \times 10 = 680$ parameters.

total number of parameters $= 832 + 1664 + 5328$
$+ 131136 + 650$
$= 137{,}610$ parameters

Task 2

Task 2a)



Task 2b)

Final accuracies

Train	Validation	Test
0.8293	0.7184	0.7127

Epoch: 17

Task 3

Task 3a)

Model 1

```
Optimizer : SGD
Learning Rate: 1e-2
Data Augmentation:  Normalization with mean = (0.5, 0.5, 0.5) and std = (0.25, 0.2
5, 0.25)
Batch Size: 64
Epochs: 10
Early Stop: 10
```

Layer Num	Layer Type	Number of Hidden Units / Num Filter	Kernel Size / Padding / Stride	Activation Function
1	Conv2D	32	5 / 2 / 1	ReLU
2	Conv2D	64	5 / 2 / 1	ReLU
3	Conv2D	128	5 / 2 / 1	ReLU
3	MaxPool2D	-	2 / 0 / 2	-
3	BatchNorm2D	128	-	-
4	Conv2D	256	3 / 1 / 1	ReLU
5	Conv2D	512	3 / 1 / 1	ReLU
6	Conv2D	1024	3 / 1 / 1	ReLU

Layer Num	Layer Type	Number of Hidden Units / Num Filter	Kernel Size / Padding / Stride	Activation Function
6	MaxPool2D	-	2 / 0 / 2	-
6	BatchNorm2D	1024	-	-
-	-	-	-	-
7	Flatten	-	-	-
8	Linear	128	-	ReLU
9	Linear	128	-	ReLU
10	Linear	10	-	Softmax

Idea: Replicating a Residual Block's use of sequential convolution layers to extract more complex features, together with the idea of using different sized kernels to extract features of different sizes from InceptionNets. Decided to exclude skip connections as the model is relatively small and the use of BatchNorm2D can reduce the impact of exploding / vanishing gradients.

Early stop at 8 epoch with 0.81 test accuracy, 0.812 validation accuracy.

Model 2

```
Optimizer : SGD
Learning Rate: 1e-2
Data Augmentation: Normalization with mean = (0.5, 0.5, 0.5) and std = (0.25, 0.2
5, 0.25)
Batch Size: 64
Epochs: 10
Early Stop: 10
```

Layer Num	Layer Type	Number of Hidden Units / Num Filter	Kernel Size/Padding/Stride	Activation Function
1	Conv2D	32	3 / 2 / 1	ReLU
1	MaxPool2D	-	2 / 0 / 2	-
2	Conv2D	64	3 / 2 / 1	ReLU
2	BatchNorm2D	64	-	-
3	Conv2D	128	5 / 2 / 1	ReLU
3	MaxPool2D	-	2 / 0 / 2	-
4	Conv2D	256	5 / 2 / 1	ReLU
4	BatchNorm2D	256	-	-
5	Conv2D	512	7 / 2 / 1	ReLU
5	MaxPool2D	-	2 / 0 / 2	-
-	-	-	-	-
6	Flatten	-	-	-
7	Linear	64	-	ReLU
8	Linear	10	-	Softmax

Idea: Alternation of MaxPool2D and BatchNorm2D after Conv2D layers. Incremental size of kernels from 3, 5 to 7 for every 2 layers. Intuition is to build more complex features based on simpler features described by smaller kernels (E.g. lines, curves). MaxPool2D to reduce noise and BatchNorm2D to prevent issues of vanishing / exploding gradients.

Stops at 10 epochs with 0.808 test accuracy and 0.817 validation accuracy.

Task 3b)

Include final accuracy scores and plot for two models

Model 1 Final Scores

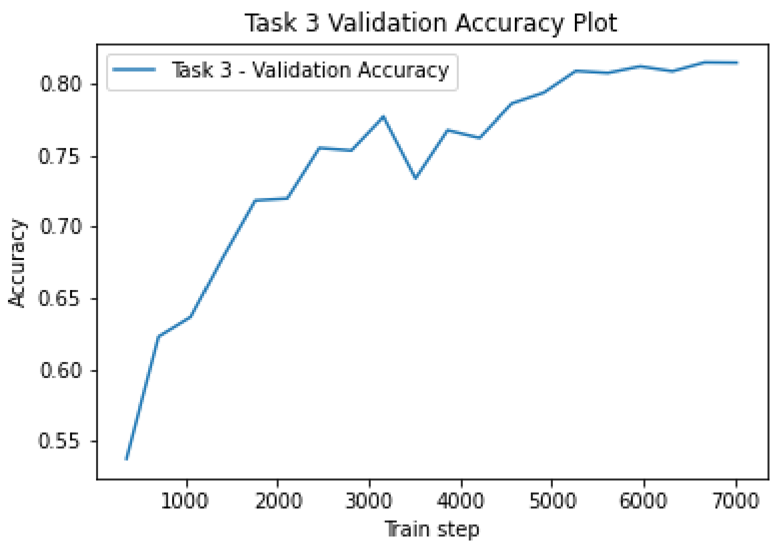
Train Loss	Train Accuracy	Validation Accuracy	Test Accuracy
0.0007	1.0000	0.8096	0.8046

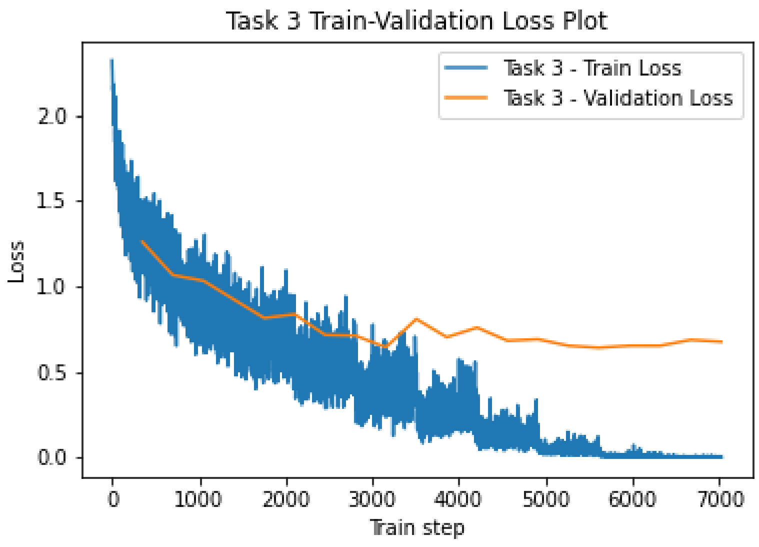
Model 2 Final Scores

Train Loss	Train Accuracy	Validation Accuracy	Test Accuracy
0.0033	1.0000	0.8146	0.8114

Conclusion: Model 2 is better marginally by 0.7%

Model 2 (Best)





Task 3c)

1. Order of Convolution Layers
    - Saw improvements when convolution layers were sequential before a MaxPool layer such as in Table 1
    - Performance improved from 0.71 to 0.75.
    - This could be because more complex layers could be detected before MaxPooling which might have removed such information.

2. More Convolution Layers
    - Saw significant improvements by simply increasing the number of convolution layers that the model has.
    - This could be due to the model being able to identify more complex features by combining features identified at earlier in the network.
    - This increased performance the most from 0.75 to 0.81.

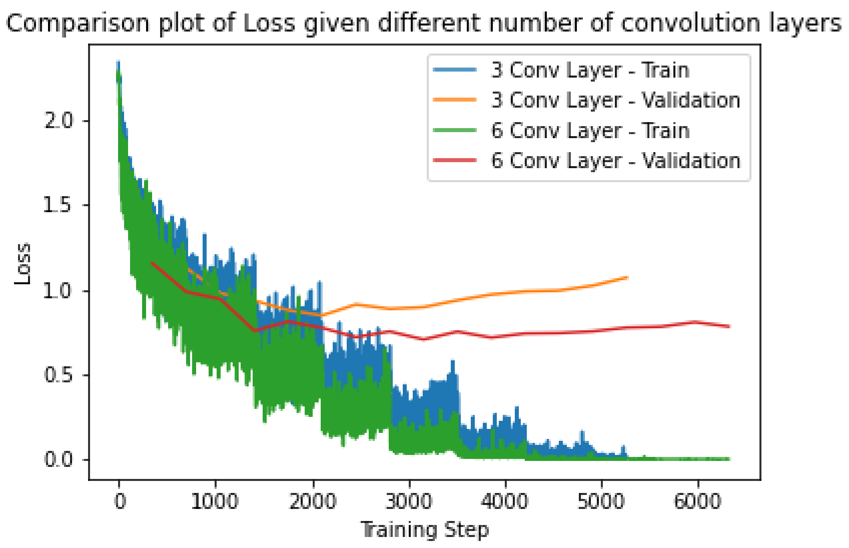
3. Increasing number of hidden units at FC layer / Increase the number of FC layers.
    - No significant improvement
    - Perhaps there isn't much information loss at this layer and more information / patterns can be extracted from the convolution stack instead.

4. Implementation of Dropout Layers at the FC layer
    - Did not see significant improvements, except that the model took longer to converge / early stop.
    - Perhaps this is due to dropout layers forcing the model to find a more reliable pattern by reducing reliance on specific nodes. The model could have insufficient complexity to do so and hence no significant performance gain was realized and the degree of overfitting stayed the same.

5. Large convolution layer with kernel size == size of input to this layer.
    - No significant improvements
    - Perhaps the FC layer is sufficient to perform such a task, albeit at a higher computation cost with less shared weights.

6. MaxPool2D first, or BatchNorm2D first?
    - Found that MaxPool2D first gave a 1-2% accuracy increase over 5 runs.
    - May be inconclusive as to which is better.
    - Order outlined in [this paper (https://stackoverflow.com/questions/42015156/the-order-of-pooling-and-normalization-layer-in-convnet)](https://stackoverflow.com/questions/42015156/the-order-of-pooling-and-normalization-layer-in-convnet) explains that BatchNorm2D first might lead to some nodes not activating when they should due to the normalization function.

Task 3d)



We saw improvement of accuracy from 0.75 to 0.81.

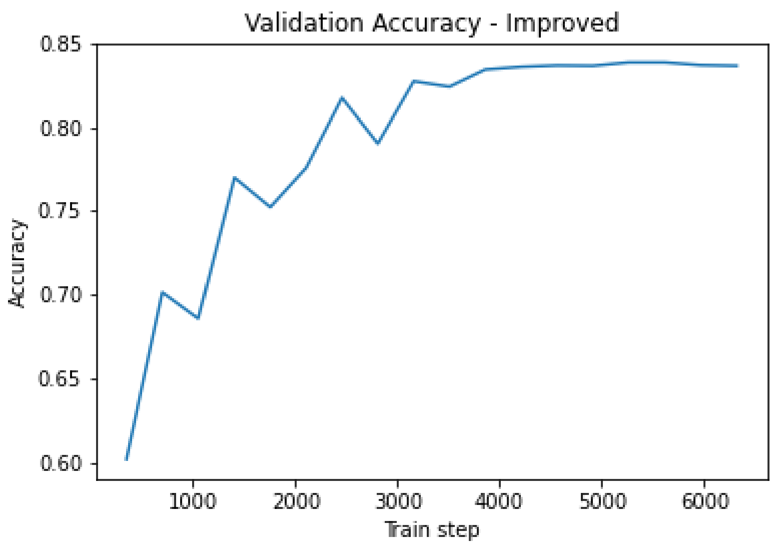
Task 3e)

Model 3

```
Optimizer : SGD
Learning Rate: 1e-2
Data Augmentation: None
Batch Size: 64
Epochs: 10
Early Stop: 10
```

Layer Num	Layer Type	Number of Hidden Units / Num Filter	Kernel Size / Padding / Stride	Activation Function
1	Conv2D	32	3 / 1 / 1	ReLU
2	Conv2D	64	3 / 1 / 1	ReLU
3	Conv2D	128	3 / 1 / 1	ReLU
3	MaxPool2D	-	2 / 0 / 2	-
3	BatchNorm2D	128	-	-
4	Conv2D	256	5 / 2 / 1	ReLU
5	Conv2D	512	5 / 2 / 1	ReLU
6	Conv2D	1024	5 / 2 / 1	ReLU
6	MaxPool2D	-	2 / 0 / 2	-
6	BatchNorm2D	1024	-	-
-	-	-	-	-
7	Flatten	-	-	-
8	Linear	64	-	ReLU
10	Linear	10	-	Softmax

Idea: Incorporating ideas from `Model 1` and `Model 2` , I order the kernels with smaller sizes first to try and get smaller features, while using larger kernels later to develop more complex features. I also used padding to keep the image the same size to allow longer networks.



Final test accuracy: >=0.83

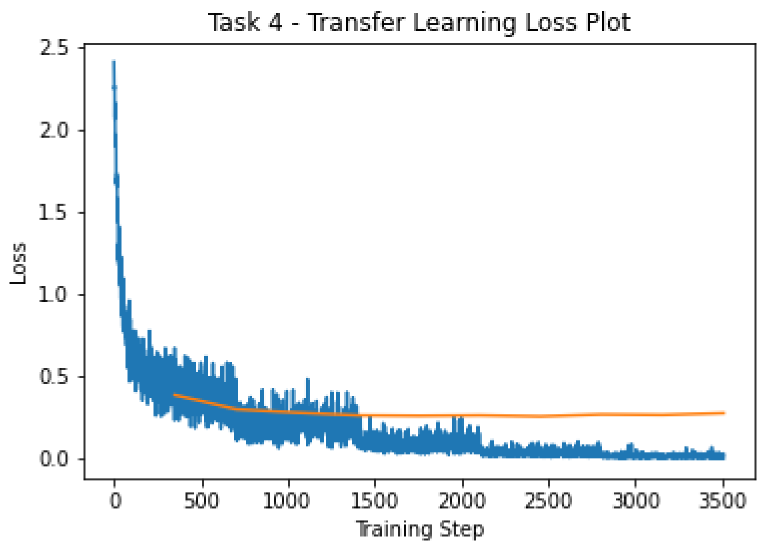
Task 3f)

Yes there are signs of overfitting. The training accuracy has hit 1 but the validation and test accuracy is still at 0.83, indicating that the model has overfitted.

Task 4

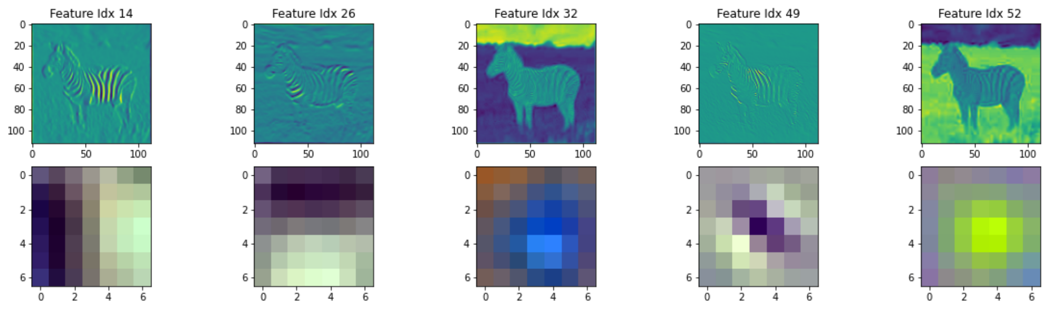
Task 4a)

```
Optimizer: Adam
Batch Size: 32
Learning Rate: 5e-5
Data Augmentations: Resize to (224, 244) and Normalize with mean = [0.485, 0.456,
 0.406] and std = [0.229, 0.224, 0.225]
Epochs = 5
Early Stopping Count = 4
```



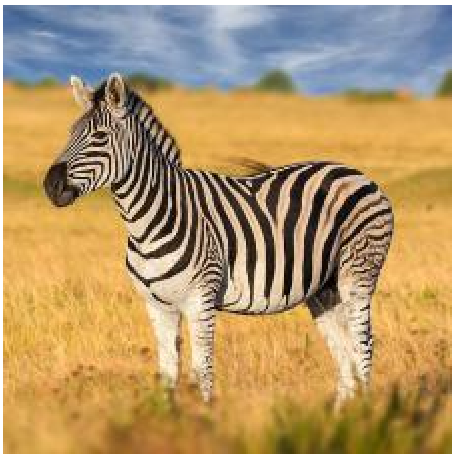
Final Test Accuracy: 0.907

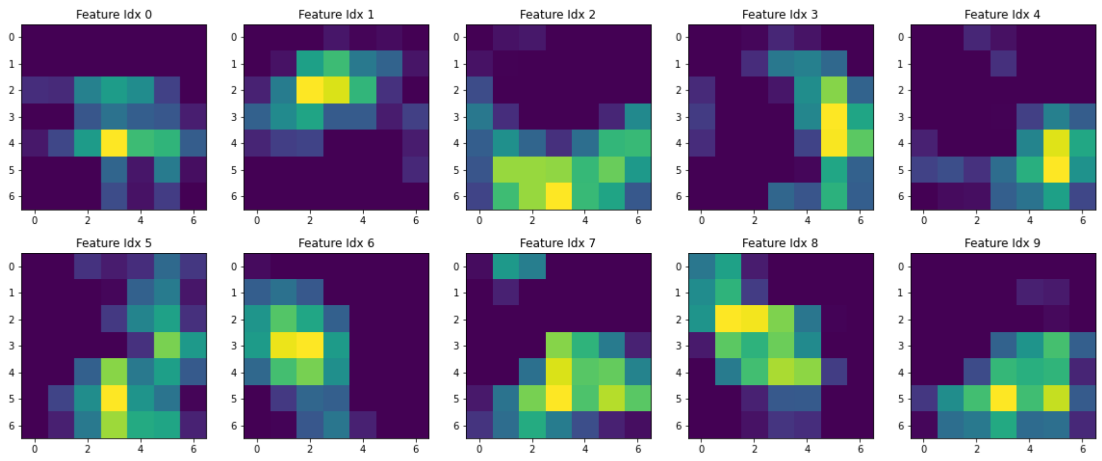
Task 4b)



The activation seems to be rather general. Some seems to have activations at concentrations of different colours. The first two also looks as if it is trying to find lines across the center, vertically and horizontally respectively.

Task 4c)





It seems that the activations largely occur where the zebra is in the picture. Some may be the head, the body or the legs.

```
In [ ]:
```