IFT6135 Representation Learning (Programming Assignment 2)

Names: Hossein Askari, Issac Sultan **Due Date:** March 9th 2018

Please find the code for Problem1 and Problem2 in this GitHub repository.

1 Regularization : weight decay, early stopping, dropout, domain prior knowledge

1.1 Overview

The experiment tested an MLP and a CNN, under multiple configurations and hyper-parameter settings:

question	model	dropout	lr0	batch size	epochs	weight decay	batch norm
Q1	MLP	false	0.02	64	100	0	false
Q2	MLP	false	0.02	64	100	2.5	false
Q3	MLP	true	0.02	64	100	0	false
Q4	MLP	true	0.02	64	100	0	false
Q5	MLP	true	0.02	64	100	0	false
Q6	CNN	false	0.0001	128	10	0.0005	true
Q7	CNN	false	0.0001	128	10	0.0005	false

Figure 1: All models were initialised with Xavier initialisation scheme

1.2 Visualisations

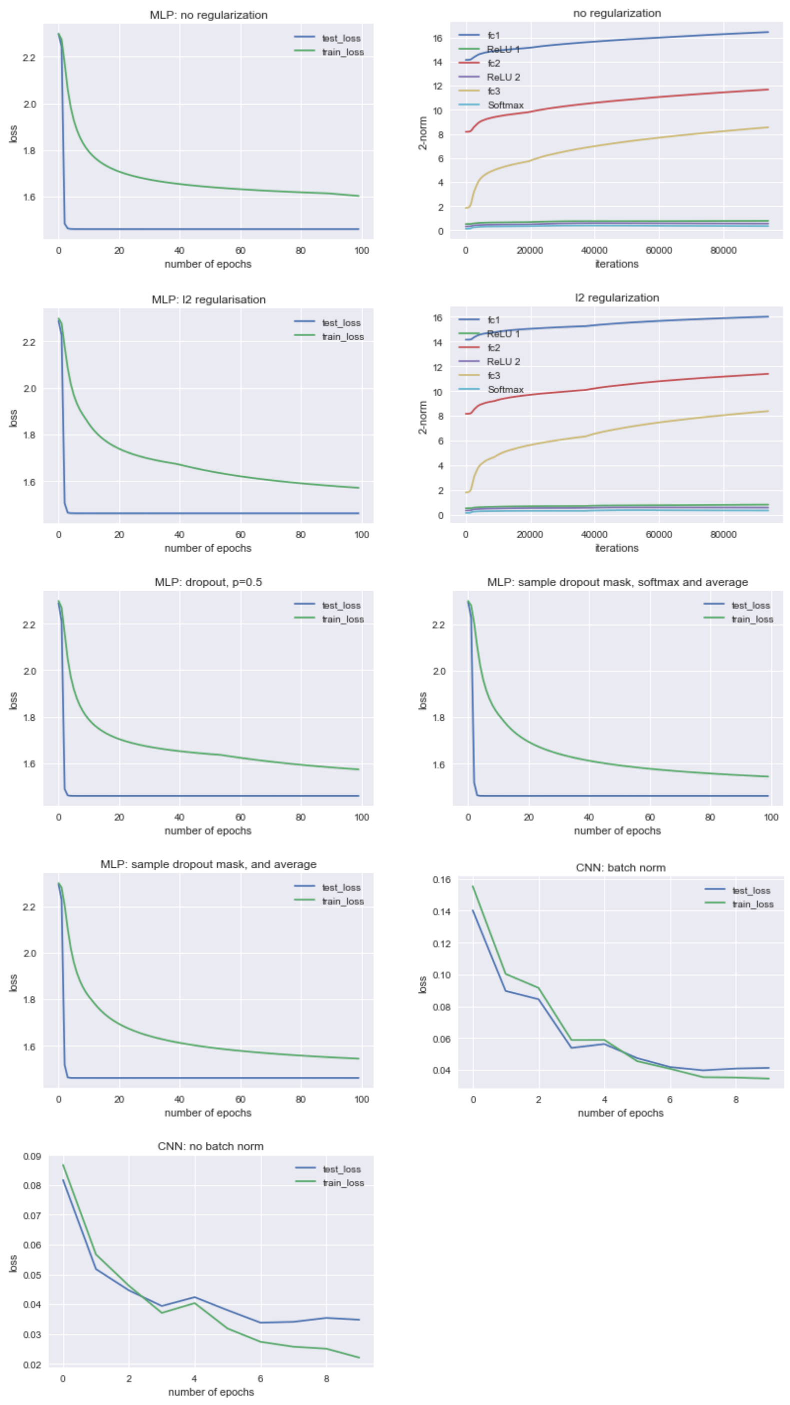

Figure 2: The learning curves, and parameter norms of the MLP and CNN models

1.3 Results

The visualizations demonstrate that:

1. Small amounts of L2-regularisation improves model performance by allowing the model to train with more iterations before signs of overfitting.
2. Similarly, dropout in the MLP model and batch-normalisation training were effective regularisers.
3. The CNN is a more outperforms the MLP in the classification of images - its hierarchy better reflects the translation-invariant nature of images.

2 Dogs vs. Cats Classification

For this part of the assignment, we were asked to train a model that can classify the Dogs Vs Cats dataset. After reading papers and reviewing methods that we learned in the class, we came up with the following architecture. In the following subsection, we describe the process of designing this model. We will then provide results on how accurate our model was able to classify the dataset.

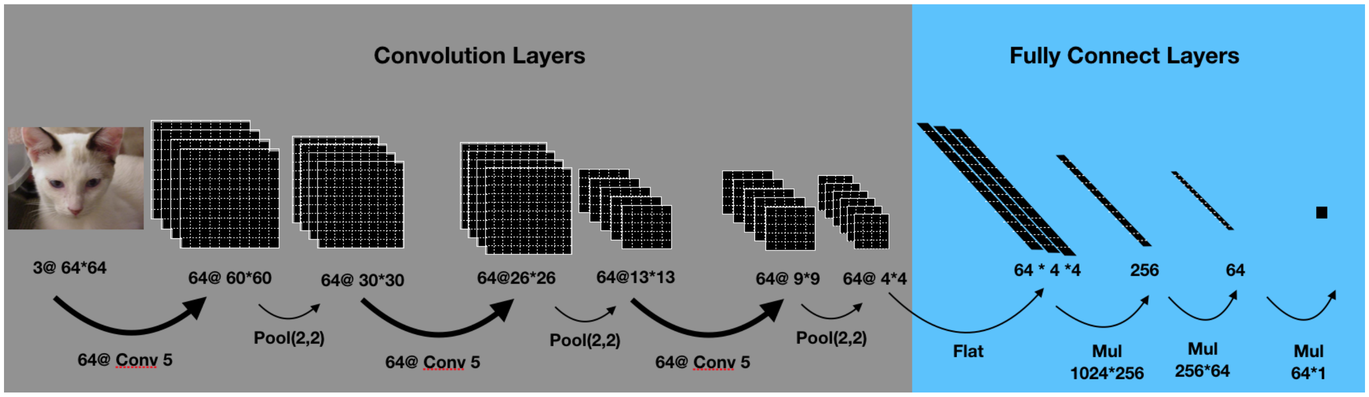

Figure 3: Model architecture for classifying Cats and Dogs data set. The model shows three layers of convolution and 3 layer of fully connected layer. The output is a single bit where 1 denotes Dog and 0 denotes a Cat.

2.1 Part A: Model Architecture

We used the model illustrated in Figure 3. As it can be seen, the model is divided into two types. The following is the model architecture:

Layer Number	Type	Size	Pooling	Batch Norm
Input	input	$3\ 64 \times 64$	No	No
1	Convolution	$64\ 60 \times 60$	Yes	Yes
2	Convoluton	$64\ 26 \times 26$	Yes	Yes
3	Convolution	$64\ 9 \times 9$	Yes	Yes
4	Fully Connected	1024	No	No
5	Fully Connected	256	No	No
6	Fully Connected	64	No	No
Output	output	1	No	No

Table 1: CNN model that was used for Cats Vs Dog classification

Table 1 shows the details of the CNN model used for Cats and Dogs classification project. As it can be seen, we used convolution layers to extract the spatial information of the image in the first layers and fully connected layers in the output to reduce dimension for the binary classification. After every convolution layer, we applied a pooling layer to reduce the input for the next layer. As it will be describe later, the batch normalization was used at every layer which resulted in shorter training time. For convolution layers, we used a kernel of size $64\ 5 \times 5$ and for pooling we used pytorch $MaxPool2d$ with a kernel of size $2\ \times 2$.

2.2 Part B: Results

We did experiment with different optimizer available in the Pytorch library. We also did experiment to find the effect of using a normalization method on training time. Figure 4 shows the training and validation curve for Cats and Dogs classifier. In this setup, we used Adam optimizer and used learning rate of 10^{-4}. We ran the model 40 times (40 epochs) over the entire dataset with minibatch of 16. The samples are taken every samples and this will give us 475 samples to plot. As it can be seen in this figure, the training error is around 2% and validation error is around 20%. These hyper parameters gave us the best model was our best performance

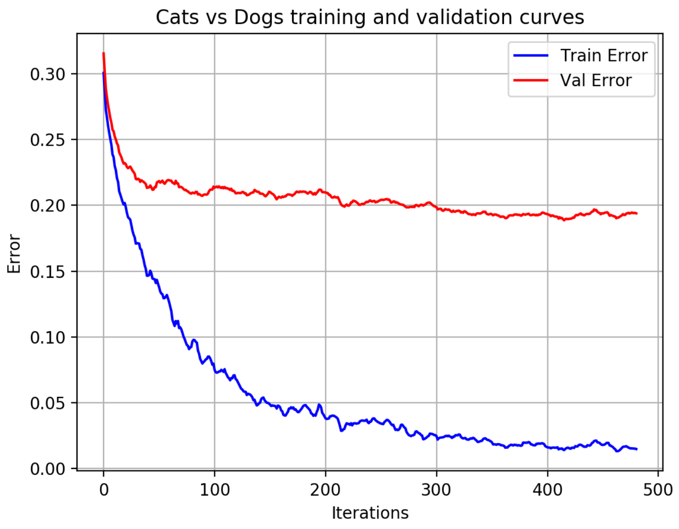

Figure 4: Training and Validation curves for Cats Vs Dogs using Adam optimizer with learning rate = 10^{-4}, number of epochs = 40, minibatch = 16 and using Batch Normalization

Figure 5 illustrates the training and validation curves for Cats Vs Dogs classification using SGD optimizer. In this setup, we used learning rate of 10^{-4} with momentum of 0.9. We ran the 40 times over the entire dataset and we used minibatch of 16. Same sampling method was used for data monitoring. In this setup we also used Batch Normalization on parameters. As it can be seen, in comparison to Adam optimizer, SGD takes longer to train. As an example, in SGD iteration 100, the error is about 15%. However, in Adam at the same iteration, the error is around 7% which is almost half. This result matches with our expectation.

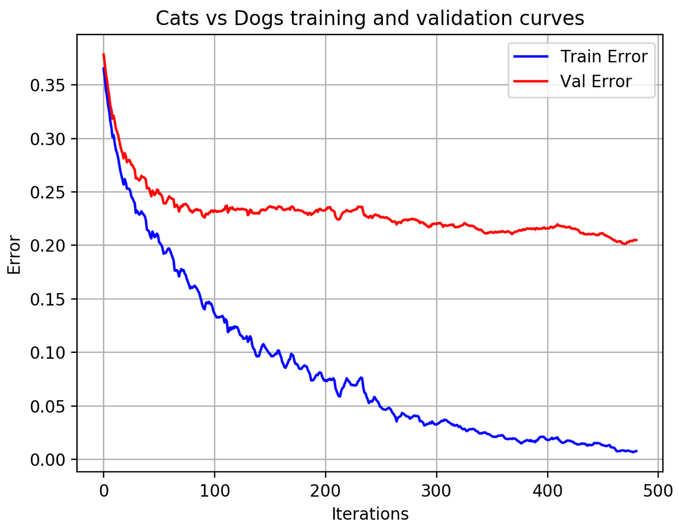

Figure 5: Training and Validation curves for Cats Vs Dogs using SGD optimizer with learning rate = 10^{-4}, momentum=0.9, number of epochs = 40, minibatch = 16 and using Batch Normalization

Finally, we did try to train the model without batch normalization using SGD optimizer. As it can be seen in Figure 6, we were unable to properly train the model.

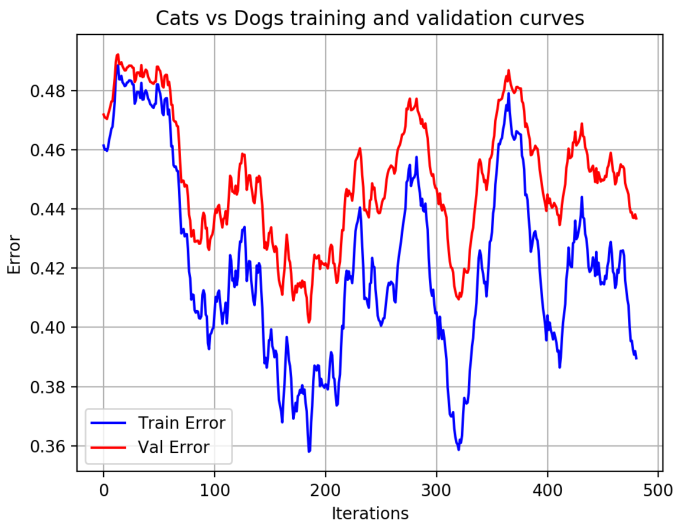

Figure 6: Training and Validation curves for Cats Vs Dogs using SGD optimizer with learning rate = 10^{-4}, number of epochs = 40, minibatch = 16 and without using Batch Normalization

2.3 Part C: Visualization

In this part we try to show what really happens when training a CNN. We decided to implement a sort of attention mechanism to illustrate how the model identifies a Cat image from a Dog image. The following figure shows this:

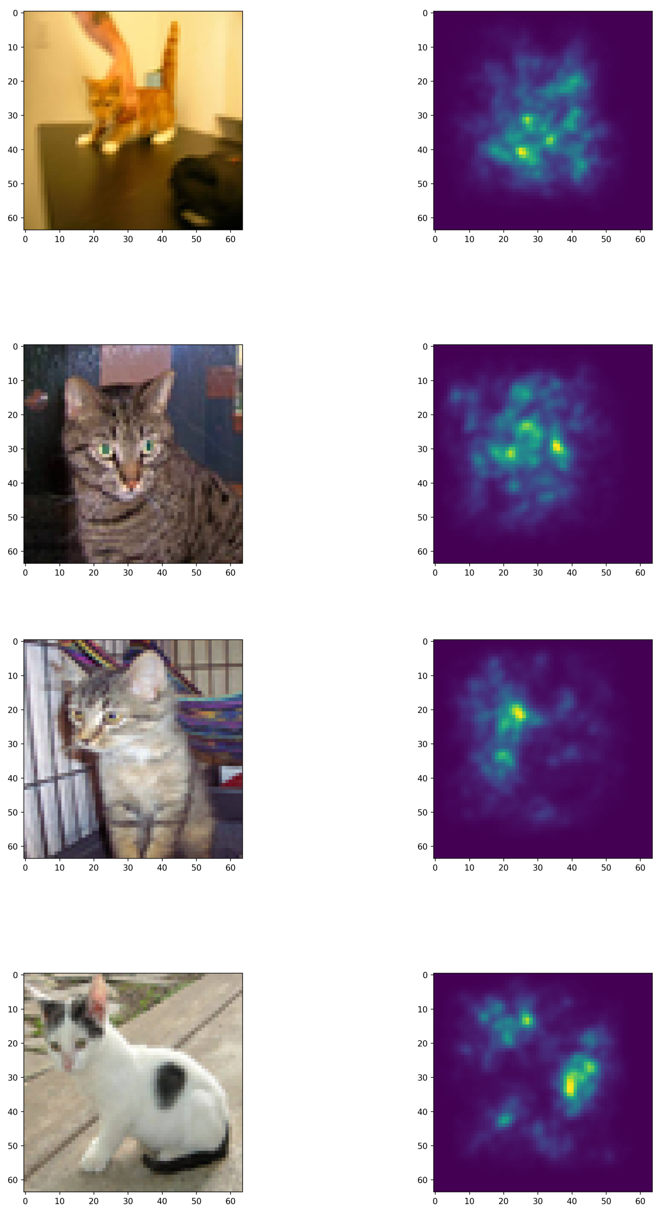

Table 2: Figure on the left shows the input cat images and figure on the right shows places where the gradient is higher showing the attention of model

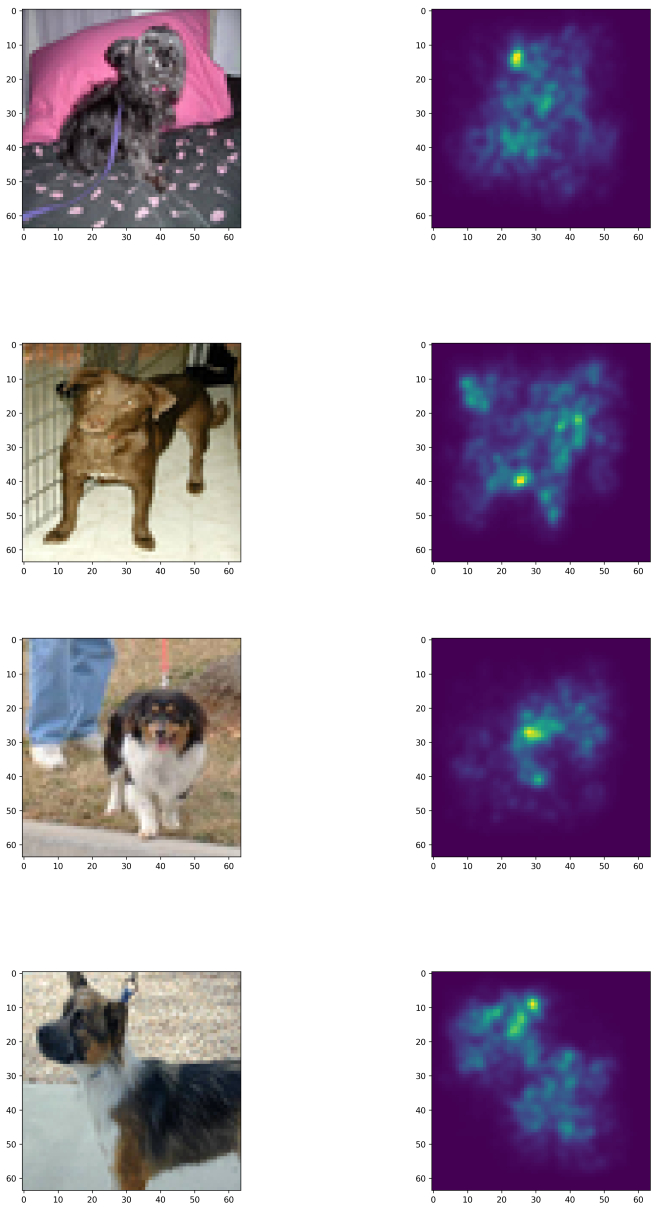

Table 3: Figure on the left shows the input Dog images and figure on the right shows places where the gradient is higher showing the attention of model

As it can be seen, the model tries to identify the faces and their shapes. The Figures on the right clearly shows that place of eye mouth and ears are very important for the model to identify a Cat from a Dog.

We have also found some interesting results on the images where the model failed to classify. For instance on the following image:

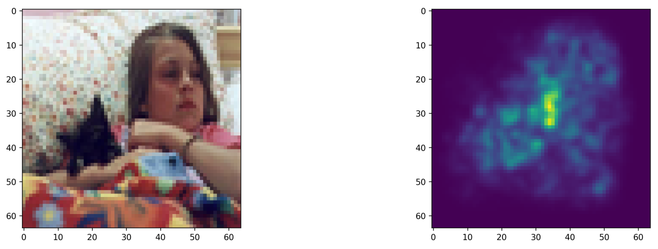

Table 4: Miss classified image by the model

Figure 4 shows an example where the model was unable to classify the image. As it can be seen, the model was unable to identify the black cat in the bed and focused on the girl's bracelet instead.

On the other hand, we found that there are some cases that the image contain both a cat and a dog. As an example, Figure 7 illustrate such scenario.

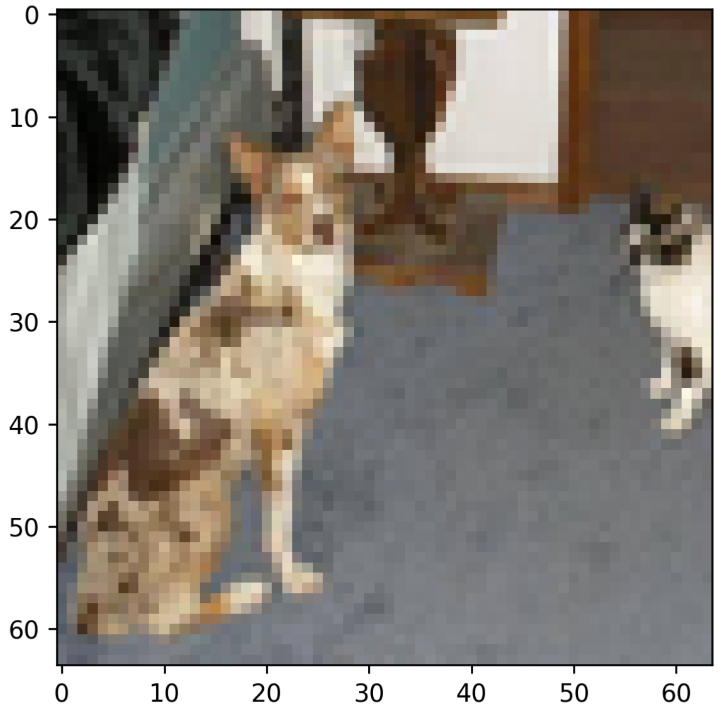

Figure 7: A case where the model classified the image with probability of 0.56

Finally, we think our model still can preform much better in a case where we do not use batch normalization. As discussed in the class, bigger model tend to learn better so we think by increasing the model size we can train our model to classify fairly good without batch normalization.