CS335 - Assignment 5

Niraj Mahajan
180050069

Task 1.3.1 - XOR

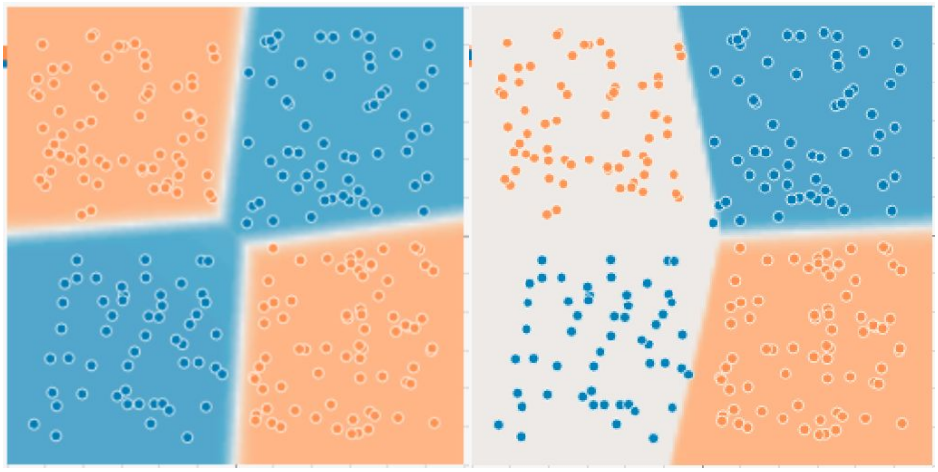
I used the following architecture for XOR

```
FullyConnectedLayer(2, 3, 'relu')
FullyConnectedLayer(3, self.out_nodes, 'softmax')
```

lr = 0.1
Batch Size = 20
Number of epochs = 30

No	Seed	Train Accuracy	Test Accuracy
1	335	98.125	97.7
2	337	98.2375	98.1
3	45	96.525	96.7
4	180050069	73.0125	73.4
5	3351	98.15	98.0
6	3352	82.675	82.3

The experimental minimal topology has 3 hidden nodes, which is a linear combination of three lines in te 2d plane. We notice that some seeds give a lower accuracy in the above topology. This is due to the initialisation of the weights which can be demonstrated in the diagram below (using the tensorflow playground tool used by sir in class). As we can see, two different initialisations produce different results. The **theoretical minimal topology** also has 3 nodes.



This drop in accuracy can be significantly reduced by increasing the number of hidden nodes to 7, but then it won't be the minimal topology that can classify a XOR dataset

Task 1.3.2 - Circle

I used the following architecture for Circle

```
FullyConnectedLayer(2, 3,'relu')
FullyConnectedLayer(3, self.out_nodes, 'softmax')
```

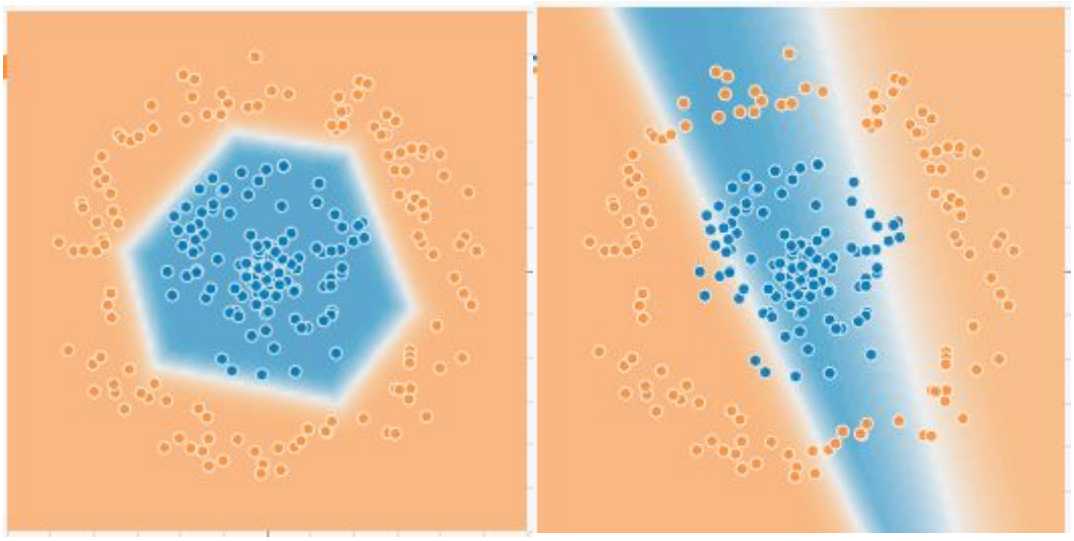
lr = 0.1

Batch Size = 20

Number of epochs = 30

No	Seed	Train Accuracy	Test Accuracy
1	335	96.6	97.8
2	337	98.475	99.4
3	45	96.525	96.7
4	180050069	99.0375	99.6
5	3351	98.375	98.3
6	3352	96.225	96.6

The experimental minimal topology has 3 nodes in the hidden layer. This combination of three hidden layers leads to an isolated polygon in the central region of the circle, as depicted in the figure below. This leads to a pretty good classification accuracy and is as good as a minimal topology. A node in the hidden layer represents a line in the 2d plane. Ideally, in order to classify 100% of the data correctly, we need to create a classifying circle which is a set of infinite tangential lines. Hence in order to achieve 100% training accuracy for any draw of a sample from the circular distribution, we will need infinite nodes. But in our case, since we have finite samples drawn, and since we allow a margin for error, good results can be obtained using just 3 nodes. In case of 2 nodes, we get decent results (above 90%), but this model leads to some elements outside the circle being wrongly classified (as in the figure) (This is logically and conceptually wrong). Hence the theoretical minimal topology is also 3 nodes in the hidden layer.



Task 1.3.3 - MNIST

I used the following architecture for MNIST

```
FullyConnectedLayer(784, 20, 'relu')
FullyConnectedLayer(20, 40, 'relu')
FullyConnectedLayer(40, self.out_nodes, 'softmax')
```

lr = 0.1
Batch Size =50
Number of epochs = 10

No	Seed	Train Accuracy	Test Accuracy
1	335	96.624	95.63
2	337	95.958	95.00
3	45	96.342	95.49
4	180050069	96.396	94.71
5	3351	96.394	94.87
6	3352	96.468	95.13

Task 1.3.4 - CIFAR10

I used the following architecture for CIFAR10

```
ConvolutionLayer([3, 32, 32], [5, 5], 16, 3, 'relu')
MaxPoolingLayer([16, 10, 10], [2, 2], 2)
FlattenLayer()
FullyConnectedLayer(400, 256, 'relu')
FullyConnectedLayer(256, 256, 'relu')
FullyConnectedLayer(256, 10, 'softmax')
```

lr = 0.1
Batch Size = 100
Number of epochs = 20

No	Seed	Train Accuracy	Test Accuracy
1	335	71.78	45.0
2	337	66.14	42.3
3	45	66.3	44.7
4	180050069	64.32	38.9
5	3351	66.16	45.2
6	3352	66.58	43.0

The training takes around 30 seconds per epoch (overall 10 minutes). I have submitted 7 model.p files, with the naming format as model_<seed>.p, each corresponding to a seed. The file ‘model.p’ corresponds to seed 3351.

Task 2.1

The intermediate layers of the CNN are a black-box that detect several features that help in classification. The initial layers can detect low-level features like the corners, edges and colours, whereas the higher level features isolate prominent features like the wheels of the vehicle, or the body of the vehicle, etc.
CNN’s share a common set of filters which are convolved over the entire image. Hence CNNs are a space-invariant operation and can detect the same features even when spatially translocated. The hierarchical layers help CNN’s detect the same features with varying sizes. CNN’s also implement Max-pooling which helps in consolidating all the results from different spatial locations of the image. In the example shown, the intermediate layers of the CNN will try to detect distinguishing features like the chassis of the motorbike vs body of the car, size of the vehicle, etc.

Task 2.2

We can train a model similar to the YOLO algorithm. We will divide the image into a grid. In each grid we run an object detection which will give us 5 values - detection value (can be 0 which means nothing found), top right corner coordinates (x,y) rectangle length and rectangle height. We can then perform IOU, ie intersection divided by Union of the areas of all the bounding boxes detected to filter out overlaps and repetitions. We can then merge the adjacent bounding boxes to form a bigger bounding box and eventually detect multiple objects of multiple (or the same) categories in the image. The limitations of this algorithm are that it fails to detect small object that are grouped together like a flock of birds. It is not rotation invariant and fails to generate a bounding box around rotated objects. Another important limitation of the YOLO algorithm is that it underperforms as compared to the state of the art method (where we slide patches over the image and perform object detection). But since the YOLO algorithm is extremely computationally efficient, we usually prefer YOLO over bounding boxes for real time applications, as even though YOLO provided suboptimal results, it is incredibly fast.

Task 2.3

We can develop the algorithm discussed in the previous task to detect several occluded vehicles. We can train the model to detect vehicle parts in the subgrids (as opposed to the entire vehicle). For example, if a wheel is detected on the road, it is highly likely that there is a car/vehicle around those coordinates (unless ofcourse a major explosion blew off the wheels of a vehicle). We can train the model to detect and differentiate car wheels vs bus wheels or car chassis vs motorbike chassis, headlights, sirens, etc. This reduces the effect of occlusion and helps detect vehicles even with obstructions. We can further refine this method by considering the detected bounding box and running it through a siamese net to compare the bounding box with those of various sample vehicles in the dataset to give an added confidence to our classification. This will help because some vehicles follow an obvious pattern. For example, auto rickshaws in the image are all yellow-green. Fire brigades are red.
The limitations of this solution are that it is difficult to distinguish between vehicles and vehicle parts at a small scale (if the objects are tiny as shown in the 2nd figure of problem statement).Although we have used a siamese net for added confidence, this might not be full proof. But in case of larger occluded images, like the first image of task3 (the one where the cars are prominent), this method should work well. Again, this method is not rotation invariant and will fail if the vehicles are oriented wrongly/inclined.

Another method is to use the labelled train data in Task 2.2 and manually add some rectangular occlusions in the image to create an augmented dataset which will help us deal with occlusions. The limitations of this method are that we need an occlusion free dataset to start with there are very few and small datasets that are free of occlusions.