Karan Agarwalla – 180050045

Problem 1.3: CS335 Lab Problems

XOR Dataset:

Architecture:
The first layer is a FullyConnectedLayer with 2 in_nodes and 4 out_nodes with softmax activation. The second layer is also FullyConnectedLayer with 4 in_nodes and 2 out_nodes with Softmax activation which is taken as predictions for each example.

Hyperparameters:
Batch Size = 50
Epochs = 20
learning rate = 0.5
Average Accuracy: 99.216 with seeds 0-49(for seeds 0-49 accuracy > 90)

Accuracy for some seeds

Seed	0	1	2	3	4
Accuracy	99.4	99.2	97.8	99.4	99.2

The theoretical minimum topology is a single hidden layer with 2 neurons and a relu activation. Average Accuracy over seeds 0-19 with lower hidden neurons:

- 3 hidden neurons with Softmax activation in both layers: 77.99
- 3 hidden neurons with relu in first layer and Softmax in second layer: 78.04
- 3 hidden neurons with relu activation in both layers (lr = 0.1, epochs = 50): 49.59

Circle Dataset:

Architecture:
The first layer is a FullyConnectedLayer with 2 in_nodes and 2 out_nodes with relu activation. The second layer is also FullyConnectedLayer with 2 in_nodes and 2 out_nodes with softmax activation which is taken as predictions for each example.

Hyperparameters:
Batch Size = 50
Epochs = 40
learning rate = 0.2
Average Accuracy: 92.546 with seeds 0-49(for all seeds except [1, 4, 11, 13, 19, 20, 23, 28, 31, 33, 34, 37, 38, 40] accuracy > 90)

Accuracy for some seeds

Seed	0	2	3	5	6
Accuracy	98.8	96.8	96.5	98.3	97.4

Due to non-linearity in the dataset, it it not possible to separate the data without any hidden layers (assuming relu and softmax activation). Average Accuracy over seeds 0-19 with lower hidden neurons:

- 1 hidden neuron with Softmax Activation in both layers: 49.725
- 1 hidden neuron with Relu in first layer and Softmax in second layer: 70.21

MNIST Dataset:

Architecture:
The first layer is a FullyConnectedLayer with 784 in_nodes and 10 out_nodes with softmax activation.

Hyperparameters:
Batch Size = 50
Epochs = 20
learning rate = 0.1
Average Accuracy: 91.7728 with seeds 0-49(for all seeds accuracy > 90)

Accuracy for some seeds

Seed	0	1	2	3	4
Accuracy	91.84	91.73	91.78	91.57	91.62

CIFAR10 Dataset:

Architecture:
The first layer is a ConvolutionalLayer with 8 filters of size (4, 4) with stride 2 and activation relu. The second layer is a MaxPooling layer with filter size (3, 3) with stride 3. These are followed by FlattenLayer and a FullyConnectedLayer with in_nodes = 200, out_nodes = 10 and softmax activation.

Hyperparameters:
Batch Size = 50
Epochs = 25
learning rate = 0.05
Average Accuracy: 45.345 with seeds 0-19(for all seeds accuracy > 35)

The model “model.p” saved for seed = 0 and accuracy = 46.4.

Accuracy for some seeds

Seed	0	1	2	3	4
Accuracy	46.4	46.5	44.6	47.3	47.01

Problem 2: CS337 Theory Problem

Task 1

CNN-networks consists of stacks of Convolution layer and Pooling layer, and finally a Dense layer for image classification. Convolution layers provide sparse interactions and weight sharing. Sparse interactions help in ensuring that neighboring pixels are more relevant to the activations. Shared parameters in a patch ensure that pixels interact in a way invariant to location in an image. Hence, similar features at different locations in the image can be detected. Take the given dataset. Convolutional layers can help extract features of wheels, engine and headlights, irrespective of their location in the image.

Pooling layers summarize the features present in a region of feature map generated by a convolutional layer. MaxPooling in particular weeds out weaker activations which may correspond to portions irrelevant to object detection (such as background). It simultaneously helps in dimensionality reduction of the feature map and helps focus on important features. The features of Pooling layer also make the model more robust to variations in position of features in the image, providing some sort of local invariance.

The different filters of the network may correspond to different features such as car's headlight, bike's engine etc. So, for example, a car would have high activations for headlights and tires. The fully connected layer computes a weighted sum of feature map and helps decide the type of object. The softmax layer provides probabilities of different classes depending on output of fully connected layer.

The activation of the layers increases in complexity with depth of the layer. The initial layers detect small features such as lines and curves. As we go deeper, the layers combine features of previous layer to derive more complex features such as wheels. The final layers combine these complex features to detect the kind of object present in the image.

A small variation in sizes of objects can be accommodated since convolutional layer correspond to local features which would not change much. For example, a filter detecting tires would still work owing to similar local features. But for large changes in size of object relative to size of kernel, object may go undetected.

Task 2

If bounding boxes are not present in training set, then we do classification.
To detect multiple object in a neural network, we replace the softmax layer with sigmoid units which squash the inputs to a range (0, 1) for every class. Also, we shift from categorical cross entropy loss function to sigmoid cross entropy loss function. Thus, an object of a class C is present in the image with probability corresponding to the activation of the corresponding sigmoidal unit.

However, there are two limitations of the above model. The sigmoid doesn't give a probability distribution over num_classes as output but independent probabilities. The second limitation arises out of its behavior with imbalanced datasets. If the number of examples corresponding to a particular class dominates, then the neural network is biased towards predicting that class.

If bounding boxes (location of objects) are given, then we use YOLO algorithm for object detection. The biggest advantage of YOLO algorithm is its speed. First the input image is divided into a grid, say of size 3*3. Now suppose there are 3 classes for detection. Then for each grid cell, we try to predict [pc, bx, by, bh, bw, c1, c2, c3]. Here,

- pc is the probability of an object center present in the grid
- bx, by represent the location of the object center relative to the grid
- bh, bw represent the dimensions of bounding box of the object relative to dimensions of grid cell
- c1, c2, c3 represent the classes. If object is of class c2, then c2 = 1 and c1 = c3 = 0

So, the problem reduces to predicting a 3*3*8 output (8 values for each grid cell). We design a conventional Convolutional Neural Network for prediction consisting of stacks of CNN layers and Max-Pooling layers. We run both forward and backward propagation to train our model. The loss function consists of penalties for bad localisation of center of cells, inaccurate height and width, classification loss for classes and loss function corresponding to pc. We can reduce the probability of multiple objects appearing in a single grid cell by increasing the size of the grid.

However, we often land up with multiple detections of a single object. In such a scenario we use Intersection over Union (IoU) and Non-Max Suppression to get rid of the unnecessary boxes. The non-suppressed box with highest probability (at least some p) is chosen and all boxes with high IoU with the box is suppressed. This is performed repeatedly.

Thus, the model can be trained and predictions can be obtained by applying non-max suppression. This model helps us in detecting objects that are far seperated. If there are two objects with same center, then this model can detect only one of them. Also, since the model is big(has a lot of parameters) it requires a lot of data to train properly. Being a supervised learning model, it requires information about the bounding boxes in the training data.

Task 3

Assumption: Bounding boxes are given
We use a similar YOLO algorithm as above. First the input image is divided into a grid, say of size 3*3. Now suppose there are 3 classes for detection. Then for each grid cell, we try to predict [pc, bx, by, bh, bw, c1, c2, c3]. Here,

- pc is the probability of an object center present in the grid
- bx, by represent the location of the object center relative to the grid
- bh, bw represent the dimensions of bounding box of the object relative to dimensions of grid cell
- c1, c2, c3 represent the classes. If object is of class c2, then c2 = 1 and c1 = c3 = 0

However, since multiple objects might overlap, we employ the concept of anchor boxes. It helps in detecting multiple objects in a single grid by defining some pre-defined shapes called anchor boxes. Each grid cell instead of having one output, will have multiple outputs. The objects are assigned to the anchor boxes based on similarity of bounding boxes and shape of anchor box.

Suppose we have 5 anchor boxes for object detection. So, the problem reduces to predicting a 3*3*8*5 output (8 values for each grid cell and anchor box). We design a conventional Convolutional Neural Network for prediction consisting of stacks of CNN layers and Max-Pooling layers. We run both forward and backward propagation to train our model. The loss function consists of penalties for bad localisation for center of cells, inaccurate height and width, classification loss for classes and loss function corresponding to pc. However, since multiple objects may appear in a single grid and interfere with each other's activations, we may choose a smaller kernel size of CNN layers, which leads to larger grid and provides robustness to occlusion.

We may often land up with multiple detections of a single object. In such a scenario we use Intersection over Union (IoU) and Non-Max Suppression to get rid of the unnecessary boxes. The non-suppressed box with highest probability

(at least some p) is chosen and all boxes with high IoU with the box is suppressed. This is performed repeatedly.

Thus, the model can be trained and predictions can be obtained by applying non-max suppression. Since the model is big(has a lot of parameters) it requires a lot of data to train properly. Being a supervised learning model, it requires information about the bounding boxes in the training data.