CS335- ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING LAB

LAB 5 - DUE DATE: 15/12/2020

PARTH LATURIA - 180050071

Q1) LAB PROBLEM

TASK 1: XOR Dataset

The first layer is a **fully connected layer** having 2 input nodes (self.in_nodes) and 4 output nodes (self.out_nodes) with a 'relu' activation.
The second layer is also a **fully connected layer** but having 4 input nodes and 2 output nodes along with a softmax activation.
Hence, there are 2 nodes in the final output, one for the samples with xor value of 1 i.e. the samples in 1st and 3rd Quadrant and the other for the samples with xor value of 0 i.e. the samples in 2nd and 4th Quadrant.
Batch size = 20
Learning Rate (lr) = 0.1
Number of epochs = 6
Following are the Seed Values used by me while developing the model along with the corresponding test accuracies obtained for each when there are 4 hidden nodes in middle layer:

SEED VALUE	TEST ACCURACY OBTAINED (in %)
0	96.3
7	98.2
15	96.6
20	96.6
42	99.1
200	95.1
1500	95.9
3487	97.9
15000	99.1
75040	95.7
180000	99.4
1500000	97
19000999	96
190409990	99.2
1904099900	96.2

Following are the Seed Values along with the corresponding test accuracies obtained for each when there are 7 hidden nodes in the middle layer:

SEED VALUE	TEST ACCURACY OBTAINED (in %)
1	98.5
3	98.9
7	97.8
15	97.6
42	98.5
150	98.7
1500	98.4
3487	97.8
15000	97.8
65916	99.2
150000	98.3
1500000	98.7
15000000	96.8
150000000	96.5
1500000000	99.8

Observations:
When the middle layer has atleast 7 hidden nodes, then the test accuracy is > 90% for almost all the seeds. This is clearly the best or most optimal topology.
When there are 4 or 5 or 6 hidden nodes in the layer, then test accuracy is > 90% for a considerable number and variety of seed values but NOT ALL.
At sizes smaller than the above mentioned topology, say,
If we have 3 hidden nodes in the middle activation set, following are the seed and accuracy values:
1(82.4%), 10(79.6%), 42(72.9%), 420(81.4%), 4200(71.8%), 42000(84.7%), 420000(70.7%), 420000(83.1%), 48020000(84.1%), 48020000(83%)
Here, the average accuracy obtained is 79.37% that is considerably less than the threshold of 90%.

Hence, The **Minimal Topology** required for classifying this XOR dataset is:
2 Fully connected layers of size 2 x 4 and 4 x 2 i.e. there are 4 hidden nodes in between input and output.

TASK 2: Circle Dataset

The first layer is a **fully connected layer** having 2 input nodes (self.in_nodes) and 3 output nodes (self.out_nodes) with a ‘relu’ activation.
The second layer is also a **fully connected layer** but having 3 input nodes and 2 output nodes along with a softmax activation.
Hence, there are 2 nodes in the final output, one for the samples with a value 1 i.e. the samples inside the circle and the other for the samples with a value of 0 i.e. the samples outside the circle.
Batch size = 20
Learning Rate (lr) = 0.1
Number of epochs = 5
Following are the Seed Values used by me while developing the model along with the corresponding test accuracies obtained for each:

SEED VALUE	TEST ACCURACY OBTAINED (in %)
3	98.1
6	98.9
7	98.3
30	96.9
42	95.4
300	95.9
3000	97.2
49710	97.3
402891	97.5
5000000	95.3
76002003	98.9
902316306	97.7
3823104057	97.2

At sizes smaller than the above mentioned topology, say,
If we have 2 hidden nodes in the middle activation set, following are the seed and accuracy values:
1(78.5%), 34(80.6%), 100(79.1%), 600(77.7%), 648(75.9%), 1000(75.9%), 3450(78.1%), 98012(77.5%), 100000(76.1%), 10000000(76.8%)
So, average accuracy = 77.62% that is considerably less than the threshold of 90%.

If we have 1 hidden nodes in the middle activation set, following are the seed and accuracy values:
1(75.5%), 10(77.1%), 42(82.4%), 100(81.3%), 1000(81%), 10000(80.1%), 20000(82.5%), 871290(79.2%), 3401234(81.5%), 90450166(76.6%)
Here, we get an average accuracy of 79.72% which is also very less than 90%.

Hence, we put no less than 3 hidden nodes in the middle activation layer.
So, The **Minimal Topology** required for classifying this circle dataset is:
2 Fully connected layers of size 2 x 3 with relu activation and 3 x 2 with softmax activation i.e. there are 3 hidden nodes in between input and output.

TASK 3: MNIST Dataset

The first layer is a **fully connected layer** having 784 input nodes (self.in_nodes) and 15 output nodes (self.out_nodes) with a ‘relu’ activation.
The second layer is also a **fully connected layer** but having 15 input nodes and 15 output nodes along with a ‘relu’ activation. It is generally recommended to use almost the same number of hidden nodes in the network. So, I kept them more or less constant.
The 3rd and final layer is also a **fully connected layer** but having 15 input nodes and 10 output nodes along with a softmax activation.
The 10 outputs of the Neural Network are the 10 different classes each corresponding to a digit from 0 to 9.

Batch size = 50
Learning Rate (lr) = 0.1
Number of epochs = 5

Following are the Seed Values used by me while developing the model along with the corresponding test accuracies obtained for each:

SEED VALUE	TEST ACCURACY OBTAINED (in %)
1	92.97
6	92.8
40	93.64
42	93.09
900	94.23
2000	94.42
30000	93.29
450000	92.39
7810000	93.98
30561923	93.48
870245619	93.74
1234567890	94.32

TASK 4: CIFAR10 Dataset

It is difficult to obtain accuracy > 35% only by incorporating Fully Connected Layers into our Model. So, we need to add convolution and average pooling layers also into our NN architecture.

The first layer is a **Convolution layer** with input depth, input row size and input column size as 3,32,32 respectively along with 'relu' activation. The filter that operates on this layer is of size 5 x 5 with a number of filters = 32 (output depth) and a stride of 3.

These high values of stride and filter size significantly downsample the input so that there are less nodes in the further layers.

The second layer is **Average Pooling layer** having 32 input channels, each with a size 10 x 10. The filter to be applied on this layer has a size 2 x 2 with a stride of 2.

The output we get from this layer is of size 32 x 5 x 5.

To pass it into the final layer that is fully connected, we need to flatten it. Hence, we apply a **flatten layer** after the average pooling layer. This outputs a flat layer of size 800.

The final layer is a **fully connected layer** with 800 input nodes and 10 output nodes along with a softmax activation.

The final output is a prediction vector of 10 classes corresponding to the input RGB image.

Batch size = 100
Learning Rate (lr) = 0.1
Number of epochs = 20

Following are the Seed Values used by me while developing the model along with the corresponding test accuracies obtained for each:

SEED VALUE	TEST ACCURACY OBTAINED (in %)
1	44.7
5	42.8
10	45.3
42	47.9
100	45.5
1000	42.5
10000	47.4
100000	44.2
1000000	46.1
10000000	45
100000000	46

Q2) THEORY PROBLEM

Note: Here, 1,2,3 refers to the respective rows of figure 2.

TASK 1:

1st layer: This layer involves the convolution layer and is primarily based on the sparse interaction property of Convolutions. It will focus on capturing the local properties of the objects like edges (vertical/horizontal), shapes, colours, and textures. It will generate and extract category independent region proposals, say, detect the borders and curvatures of the object in the image.
On applying the convolution, the input will now get transformed in such a way that the next layers will be able to detect significant subparts of the object present
2nd layer: Application of a filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in the input by tweaking the parameters of the convolution. This layer will detect smaller subparts of the object using the previously developed database. It will try to extract sensible components from the image and pass it onto the next layer.
For 2 (bikes), this layer will detect the presence of a handle at the front, 2 wheels: one at the front and other at the back, seat in the middle, mirror, etc.
For 1 and 3 (car), it will detect the presence of wheels, bonnet, roof, dicky, number plate, windshield, headlights, etc. Here, both the images will have very similar output due to the presence of shared parameters. Since, the same parameters are used all over the image, a similar operation will take place over different patches of the image.
It will also perform pooling to do pixel selection. This will help us filter out the redundant/noisy pixels or subparts of the image. It also helps us reduce the number of activation nodes and hence the number of parameters in further layers.
We then flatten out the output of this layer and pass it to the fully connected layer (final one).
Final layer: The final layer will try to merge all the smaller detected subparts and try to determine the vehicle formed in a larger sense. It will then try to associate it with the vehicular class it resembles the most and hence classify it as a sample of that class. This layer supports the broadest and most abstract level of classification. It will ignore and overlook small outliers and trivial features in the vehicles and so, will tend to find more dominant features and classify the vehicles under one of the N classes available.
Eg: Bonnet + windshield + Dicky in 1 and 3 => 1 and 3 resemble a Car.

The optimal and efficient results of CNN are due to its following properties:
Sparse Interaction: It focuses on the fact that a pixel/data value is influenced and determined to a very large extent only by its neighbours. Hence, only their contribution should be considered. So, the effect of other pixels far away is removed. It helps us represent the local structure of a feature.
Eg: Connection between dicky and wheel in 1 and 3,Connection between headlight and handle in 2.
Weight Sharing, Equivariance: These help us become invariant to the translation effect of different subparts of the object. They also help us reduce the number of weights used for detecting the object.
Weight sharing is similar to moving patches. It also helps identify recurring components. Eg: Wheels in 1,2,3, Handle, mirrors, etc
Equivariant Representation also helps us conclude that 1 and 3 belong to the same class.
The fundamental properties of CNN like Sparse Interaction, Shared Parameters and Invariance based on Equivariance helps it become robust to normal operations on objects scaling, rotation, translation, reflection, etc that do not actually change the innate structure and feature of the object.

TASK 2:

The fundamental approach to identify different vehicular categories all present in the same image is:
We try to detect whether each vehicular class is present or not, independent from the other classes.
We try to determine the significance of a class being present in that image in an absolute sense.

The modification in our previous Network Design to support the above mentioned idea is:
We try to utilize the annotated data (available to us in the form of training data) and attempt to output the probabilities of occurrence of each class (in an absolute sense), independent from the others rather than outputting the relative chances of presence of each class. So, now, the probability values of the classes need not sum to 1. Also, each of them can unbiasedly assume any value between 0 and 1.
Then, we set a certain threshold such that all the classes that have their probability value greater than or equal to this threshold are declared to be present in the image and those with probability values lesser than it are not considered to be present in it.
Note that this threshold setting will certainly depend on how many vehicles we need to detect in the image and also on how the obtained probability values are distributed in the [0,1] range.
For this, instead of using softmax, we should be using sigmoid function. This helps us measure each label on its own merits.

Based on this, we can also output a k-hot vector instead of a 1-hot vector where k is the number of objects that are considered to be present in the given image.
Eg: say we have a total of 5 possible vehicular classes and only bike and car seem to be present in the image using our previous steps. So, the output will be: [1,0,1,0,1] where each of the two '1' correspond to bike and car respectively.

Apart from this, we can also use some complex techniques like Bounding Boxes and Image Segmentation.

The **Limitation** to the idea described above are:

- Since initial labellings and valued data points are required for the above steps to work, the presence of Annotated data is a must here.
- The above procedure must also be told about k i.e. about how many vehicles it has to detect. So, the above idea cannot just demarcate between classes with significantly high probability of occurrence and those with negligible probability of occurrence **all by itself.** So, it will require a threshold to work to our expectation.
- This idea will perform very poorly if the image consists of overlapped patches of different vehicles.
- Class Imbalance and Biased computation of the probabilities due to misunderstanding of the pixels can also be another point of concern (but not that serious)

TASK 3:

To solve this issue of Occlusion, we need to reduce the dependence of our probabilistic output on the entire object. Our model should be able to predict the correct class even from a partial portion of the object. If it does so, then we can train our model on the part of the vehicle that is not overlapping or not occluded and so can predict the output correctly even if some part of the vehicle is occluded.
To incorporate this idea into the model, we can do the following:
We try to train the model using partially occluded samples. This helps us reduce the spatial support base for the filters. Also, a regularizer that shrinks the spatial support for the filters will support the purpose.
Apart from this, we can also try to use apt sharpening filters on the non occluded portions that can magnify and enhance its features and so can give us a clear idea of the class that this particular object belongs to without actually working on the whole object.

The **Limitation** is:
We need to train our model correctly only on the part that is not occluded. This requires us to identify the portion of occlusion and so demands more localization operations. This also demands some involvement of segmentation of the input images that is considerably difficult to incorporate since it will make the model very much complex.