ML/AI Lab 5

1.3> XOR Dataset
(a)

Architecture: 1st layer is a Fully Connected Layer with 2 in_nodes and 4 out_nodes and ReLU activation.

With random seed as 42, 4 was the minimum number of hidden nodes for > 90% accuracy

With higher number of hidden nodes (eg: 6 or 7) it is more than 90% for most random seeds.

Second layer is also a Fully Connected Layer with 4 in-nodes, 2 out_nodes and softmax activation.
The 2 ~~out~~ out_nodes are the predictions.

Test accuracy with seed = 42: ~~98.8~~% 99.2% for 7 hidden nodes,
98.8% for 4 hidden nodes

#epochs = 10
lr = 0.1
batch_size = 20

With 7 hidden nodes:
seed 421 → 99.6%, seed 32 → 98.4%, seed 3 → 97.4%,
seed 4 → 98.8%, ~~seed 5 →~~ seed 9 → 98.8%.

With 6 hidden nodes also most seeds were giving > 90% accuracy but 7 had a better success rate.

Note: In my code I have used 7 hidden nodes.
So first layer has 7 out_nodes,
second layer has 7 in_nodes.

(b) Circle dataset

Architecture: 1st layer is a Fully Connected Layer with 2 in_nodes and 2 out_nodes and ReLU activation. With random seed as 42, 2 was the minimum number of nodes for >90% accuracy.

With higher number of hidden nodes (eg: 5) or 6, it is >90% for most random seeds.

Second layer is also a Fully Connected Layer with 2 in_nodes, 2 out_nodes and softmax activation.

The 2 out-nodes are the predictions.

#epochs: 15
lr: 0.1
batch_size: 20

Test accuracy with seed = 42 : 99.4% for 5 hidden nodes.
97% for 2 hidden nodes

With 5 hidden nodes:
seed:4 → 98.6%, seed 3 → 99%, seed 7 → 97.9%,
seed: 71 → 99%, seed 421 → 98.6%.

Note: In my code I have used 5 hidden nodes.
So first layer has 5 out_nodes,
second layer has 5 in_nodes.

MNIST

(c) Architechre: First layer is a Fully Connected Layer with 784 in_nodes and 15 out_nodes and ReLU activation.

Second layer is a Fully Connected Layer with 15 in_nodes, 15 out_nodes and ReLU activation.

discussed in class { (Recall : It is well known convention to keep equal number of hidden nodes across hidden layers)

Final layer is a Fully Connected Layer with 15 in_nodes, and 10 out_nodes representing 10 classes with softmax activation.

#epochs: 5
Lr: 0.1
batch_size: 50

Seed	Test Accuracy
42	93.49%
4	92.51%
5	94.14%
71	93.61%
69	92.58%

Average Test Accuracy: 93.27%

1) CIFAR10

Architecture: Input to first layer is the 3x32x32 image for each training example in the batch.

First layer is a ConvalutionLayer that takes 3x32x32 input image and outputs a deep representation of size 32x10x10 by using 96 (32x3 -out-depth x indepth) convolution filters of size 5x5 with stride as 3. Activation is ReLU. No padding

2nd layer is Avg Pooling Layer which gives output of size 32x4x4 by using an average filter of size 4x4 with stride = 2.

3rd Layer is Flatten Layer with output of 512 nodes.
Final layer is Fully Connected Layer with 512 input nodes and 10 classes as the output with softmax activation.

(10 epochs sufficient to cross 35%)

#epochs: 50
Lr: ~~0.1~~ 0.1
batch_size: 50

Seed	Test Accuracy
2	49.5%
42	50.2%
6	49.1%
5	46.8%
4	52.6%

Avg Test Accuracy: 49.64%

I have saved model.p with seed: 4.
With seed: 42 is also given.

Lab 5

Assignment 5

2) 1. Task 1: CNNs take into account the spatial structure and classify using smaller number of parameters than MLP.

The intermediate layers of the CNN will be responsible for different features of the image which will help the last softmax layer to activate a vehicular category.

We will be referring to 3 pictures in Figure 2 as ①,② and ③.

In ① for example, some activations will be responsible for identifying parts of the door, some will take care of the bonnets. There will be layers and activations corresponding to headlights, front, windows, numberplate, logo, top, wheels, windshield etc.

In ②, the features taken to care by intermediate activations will be: handle, tyres, mirrors, seats, engine-like structure visible on the side.

Note that ~~they~~ there maybe subdivisions within them also. (Learning different parts of a feature: spokes of wheels, parts of headlight, edges of windscreen, boundaries of seat etc

③ is almost same as ① modulo the shift so the identifiable features will remain same

Now, let us discuss helpful properties of CNN for this task

Note: we could have many feature maps in our network leading to multiple parallel pipelines merging at the fully connected layer. In this case, the individual pipelines maybe responsible for separate features (door/bonnet).

→ first 2 features help in distinguishing size and location of image (object)

3 main features of CNNs:

→ local connections through filters: neighboring signals more important than ones far away. It helps to represent local ~~structure of~~ structure of a feature
(Eg: connections b/w doors and wheels in ① & ③ or the connections b/w seat and engine of the bike in ②.

→ Weight/Parameter sharing: It helps in identifying recurring components (Eg: wheels, headlights, doors, mirrors etc.)
It can be visualized as moving patches.
The intuition here is that neighbouring signals/pixels affect in a similar way of location.

→ Equivariant Representation: $f(g(\cdot)) = g(f(\cdot))$
where f is the convulution and g is the shift.
This in particular helps to identify ① and ③ as the same class since our CNN is robust to shifts

Some other properties like max pooling provide a sharpening effect and will help to avoid the noise/background as in ②. It will help ignore the road from the bike image.
It helps us to pick peak of many signals and is locally translation invariant.
Strides help in down sampling and moving patches at regular frequencies.

2. Task 2 : A naive approach would be to classify parts of the image independently.

Since we have annotated data, one thing we can do is - if we are expecting more than one object in the same image, we can have the model output the probabilities of each class, rather than just outputting the predicted class. Then, the idea is that we can filter out very low probabilities and keep those scores above a certain threshold.

So, instead of using Softmax, we should use sigmoid for our final layer, so that each label is measured on its own merits (and not compared against its neighbours).

Other technique we can have labels with more than 1 1 in a row.

eg: [car_label, bike_label] ∈ {[0,0], [0,1], [1,0], [1,1]} instead of only [0,1] or [1,0].

Some other complicated techniques exist like image segmentation and bounding boxes. (Algorithms like YOLO {You only Look Once} and SSD [Single Shot Multibox Detector})

They look at images and predict bounding boxes for the classes. Since the classes are separated we can use this.

"Image segmentation" which classifies on a per-pixel basis can also be used. We may use pretrained models to first segment objects in the scene & then filter out object regions using those segmented labels.

(This has limitation of "per-pixel".)

Limitations:

The most important limitation is that we need annotated data. Each image should preferrably be labelled with the objects its contained. The network, if possible can be trained with some multi-vehicle images.

Setting the threshold in my first method is another concern. (Hyperparameter optimization may have to be used).

This network will perform poorly on overlapped images.

Class Imbalance is also a serious issue since it will be better if images with well separated vehicles can be fed to the network during training else it might always predict one class.

We may also use R-CNN model (Regions with CNN features). which takes different regions and predicts class for the regions.

Limitation is that since it is 2-stage (detect region then learn), it cannot be implemented real time. The search region time could lead to generation of bad candidate proposals since no learning happening at that stage

Cropping may lead to loss of information

It is complex (R-CNN) since we extract many regions for each image and the no. of CNN features also increases drastically.

3. <u>Task 3</u>: The condition mentioned is called occlusion.

We could do 2 things:- 1 is train the classifier using partially occluded object examples.

Research has been done in this area and it ~~strong~~ shows that such training reduces spatial support of filters. Also, we can have a training process that uses regularization to shrink spatial support for filters. By smaller spatial support, we mean small support relative to filters in networks trained as usual.

An occlusion robust classifier should use features that do not rely on spatial support of entire object but only a part of it. If it does so, it can be robust without being trained on ~~occ~~ occluded objects

Limitation: May be it will not handle all aspects of occlusion. We also need some localization to handle occlusion.

Returning to the first method, we could use dataset augmentation. But if we just change the dataset the accuracy will degrade with testing on high occlusion. So we can also do something else

Integrating novel preprocessing stages to segment the input and impaint occlusions can help. A CNN so modified will be effective even with high occlusion

Such a network will also be more accurate on unoccluded images. These results depend on successful segmentation. A good dataset would be where occlusions are easy to segment from figure and background.

(Limitation): Achieving similar results on more challenging datasets would require finding a method to split figure, background, and occluding pixels in the input.