CS335 Assignment 5

Anish Deshpande: 180100013

XOR:

FullyConnected(2,4, relu)
FullyConnected(4,2,softmax)
Seeds:[10,15,19,20,22]
Average test accuracy = (95.7+97.5+95.9+95.1+98)/5 = 96.44%
Average test accuracy with 3 neurons (4-1) in the hidden layer, same seeds = (82.2+64.9+73.6+70.6+84.2)/5 =75.1%
Hence, our structure is optimal.

Circle:

FullyConnected(2,2,relu)
FullyConnected(2,2,softmax)
Seeds:[14,16,18,21,22]
Average test accuracy = (96+98.8+95.4+96+98)/5 = 96.84%
Average test accuracy with 3 neurons (4-1) in the hidden layer, same seeds = (80.6+81.5+79.6+76.5+77)/5 = 79.04%
Hence, our structure is optimal.

MNIST:

FullyConnected(784,16, relu)
FullyConnected(16,16,relu)
FullyConnected(16,10,softmax)
Seeds:[22,23,24,25,26]
Average test accuracy = (91.24+90.9+90.58+90.61+90.7)/5=90.81%
Hence, our architecture is satisfactory

CIFAR:

ConvLayer([inp.shape],[5,5],32,3,relu)
AvgPoolingLayer([32,10,10],[4,4],2)
FlattenLayer()
FullyConnectedLayer(softmax)
Seeds:[14,15,23,24,25]

Date :

Assignment 5

Task 1

* In general, the first layers of a neural network identify low-level features like lines/edges. The next layers take combinations of these outputs to identify more abstract objects, like shapes. And the final layers reach the highest level of abstraction to actually identify the desired objects.
* So, for row 1 and row 3 of fig. 2 (the cars), the intermediate feature maps detect the (same) similar features of the car, but at different locations (translation). This is sensitivity that can make our network go wrong.
* The pooling layers help in increasing the robustness of the network. for example, in Max Pooling, the most 'activated' feature in each patch is collected and passed forwards. This dowsampling will help achieve translation invarience on the identifying features that make up the car, but it will lose the relative position details of the features. (So, a car broken up into pieces may still be recognised as a car).
* Weight sharing in the convolutional layers helps. achieve translational invariance too.
* The bike has a different set of features, so the different activations in the network will help us distinguish it

Date :

from the car.

* The use of filters which are scale invariant and using pooling layers to downsample the relevant features helps to tackle identifying ~~the~~ the objects of the same type but different sizes.

Task 2

i) * One simple way is to divide the image randomly into a few parts of reasonable size, to see whether any sub-image (after appropriate resizing) detects an object. But this is naïve.

ii) * One way is to have the output of the network be a probabilistic model of all classes, and allow for multiple predicted classes for the same input example by putting a minimum threshold on the probability required to identify any class as present. (sigmoid)

iii) * If the number of possible objects is known (eg: car and bike), then we can stack two CNNs in parallel, into one network, with each being trained to identify only whether a certain kind of object is present or not (one vs rest type). So, if the input has multiple types of object, then the CNN's respective parts will get activated and ~~and~~ detect the presence of (both) all of the objects. This is more computationally expensive, but not as bad as multiple sequential CNNs.

iv) * Point (ii) implies the use of sigmoid instead of softmax, or something similar.

Date : ______

* An important limitation for the method in point (iii) is computational cost and time.
* A limitation of the method in point (ii)/(iv) is that we may lose inter-dependence between objects, which may be desired, leading to more false positives (eg: An increase in the probability of a car being present does not decrease the probability of a bike's presence).

* Another method is to look at multiple CNN layers to detect an object. The activations of one object may begin to override those of another in the later layers. So, we use outputs from some select convolutional layers directly in a layer further down the network. This way, we make use of features while they can still be resolved, before losing them forever.

Task3:

* To handle occlusion, our filters shouldn't be too large spatially. They should be small enough to be completely covered by one part of an object, without spilling over into the neighbouring overlapped object.
* So, the feature detected here belongs entirely to one class.
* If we have, a CNN trained on well separated images, then the feature of the partially-occluded object we capture will help us match it with the type of object it is, based on the different features our network extracts from each object.
* We can add some regularisation term to shrink the kernel/filter support (spatially), and this will help us detect small or partially visible features of an object.
* A limitation of this is that some essential features which are large in size for a particular type of object may not be able to be captured.
* Also, well-separated images may not always be available to help in training, so in that case, the ~~t~~est accuracy will fall (training only on occluded images).