1.3) The topologies for the different datasets used are:
MNIST:
A neural network with one fully connected layer with 100 nodes and one output layer(fully connected layer) with 10 nodes. The hidden layer has relu as activation function and output layer has softmax as activation function.

CIFAR-10:
A neural network with one convolutional layer having input size as (3,32,32), filter size as (4,4), number of fiters as 32, stride as 2 followed by a max pooling layer having filter size as (3,3) and stride as 3 followed by a flatten layer followed by a fully connected layer with 45 nodes and finally an output layer(fully connected layer) with 10 nodes. The convolutional layer and the hidden fully connected layer have relu as activation function and the output fully connected layer has softmax as activation function.

XOR:
A neural network with one fully connected layer with 4 nodes and one output layer(fully connected layer) with 2 nodes. Both the hidden layer and output layer have softmax as activation function. The output has 2 nodes because the classification is binary. We need 4 nodes in the hidden layer because we need to classify data in four quadrants with softmax.

The test accuracies when we remove one node from the hidden layer for different seeds are:
335 – 71.6
193 – 85.9
225 – 82.39
42 - 82.8
97 - 84.39
average = 81.42

CIRCLE:
A neural network with one fully connected layer with 3 nodes and one output layer(fully connected layer) with 2 nodes. Both the hidden layer and output layer have softmax as activation function. The output has 2 nodes because the classification is binary. We need 3 nodes in the hidden layer because we need to represent a circle with straight lines to classify the data in the inner circle. A circle can surrounded by a polygon when it needs to be represented by straight lines and since, the minimum number of lines needed for a polygon is 3, we need atleast 3 nodes in the hidden layer.

The test accuracies when we remove one node from the hidden layer for different seeds are:
335 – 81.39
193 – 79.5
225 – 80.9
42 - 82.3
97 - 79.3
average = 80.68

The test accuracies along with the seeds used for different data sets for the above topologies are as given:

MNIST:
225 - 96.47
193 - 95.92
335 - 96.34
42 - 96.08
97 - 96.28

XOR:
97 - 99.1
42 - 98.5
335 – 99.9
193 – 99.6
225 – 98.0
average = 99.02

circle:
225 - 96.8
193 - 97.2
335 - 98.2
42 - 97.39
97 - 97.8
average = 97.48

CIFAR-10:
97 - 44.0
42 - 47.69
335 - 45.6
193 - 44.5
225 – 45.1

2)
Task1: The CNN consists of a convolutional network along with maxpool layers. The convolutional layers extract some features layer by layer and combine them to extract complicated features. For example, in the car and bike example given, the initial layer may extract the horizontal and vertical edges in the figure, the next layer extracts the curved and slanted edges. The next layer may extract the curved, slanted, horizontal, vertical edges combined as different parts. The next layer then maybe extracts colour information and then extracts the information of a part of a specific colour like round+black+small – car tyres, round+black+big – bike tyres. Thus the information like size and colours are captured through different layers and are finally used to classify them as bikes or cars.
The difference in sizes is handled by using many layers of filters of varying sizes. So, since the size of the object is atmost the size of the image, we can use filters of sizes of popularly occuring images for different layers to capture objects of varying sizes.
The difference in positions is handled by the maxpooling layers which are translation invariant. The convolution layers extract the features and the max pooling layers provide translational invariance. If we use only max pooling instead, we might end up calling jumbled parts of a car in an image together as a car. So, we need to use convolution first to extract the features and then use max pooling for translation invariance.

Figure2: a) As explained above, the initial layers extract the horizontal vertical edges, next layers extract curves and slant edges, next layers extract colours like orange and black and shapes like car body shape and circles, head light shape and the next layers extract colours mixed with shapes like black+circle, orange+car body and so on. After extracting these features in the convolutional layers, the max pooling layer takes these feature extractions and outputs a new matrix. The presence of particular values in this matrix at particular places implies the existence of car at those pixels in the original image that correspond to these elements in the matrix. This can be trained using fully connected layers after applying the max pooling.

b) As explained above, the initial layers extract the horizontal vertical edges, next layers extract curves and slant edges, next layers extract colours like black and shapes like bike body shape and circles, head light shape and the next layers extract colours mixed with shapes like black+circle, black+bike body and so on. After extracting these features in the convolutional layers, the max pooling layer takes these feature extractions and outputs a new matrix. The presence of particular values in this matrix at particular places implies the existence of bike at those pixels in the original image that correspond to these elements in the matrix This can be trained using fully connected layers after applying the max pooling.

c) The extraction of features in this image takes place exactly as in the first image, but the elements of the matrix corresponding to car in the output of maxpooling will be in different positions but have same values due to the translational invariance provided by the max pooling layer. This will not be a problem for recognition since, all we need to predict the presence of the car is the presence of those values in the input of the fully connected layer.

Task2:
To identify multiple objects in an image, we use a sliding window of different sizes and apply the image recognition of Task1 on each part of the image that is covered by the window. The window size can be some popular sizes in the image data assuming we know the sizes of objects in the data. After identifying the object in the window as some object with some probability, we choose to predict that class for the object if the probablity for the predicted class is greater than some threshold. Basically, what it means is, instead of directly giving the output of the convolution layer to the max pooling layer, give only a sliding window size of the convoluted data to the max pooling layer. So the change to the CNN network of Task1 would be to add a new sliding window layer between Convolutional layer and Max pooling layer with it's own parameters like window size and stride. Also, if we want to maintain the max pooling layer as it is, we may pad the slide window sized patch to make it's dimensions same as the output of convolutional layer.
Limitation: This method takes advantage of the fact that the objects are very well separated in the image. So, by using a sliding window and some additional training, we are converting the problem of multi object detection to single object detection, which means that it will not work well for overlapping objects. Also, it is computationally very expensive since we need to generate multiple patches from the same image for training and identifying multiple objects.

Task3:
To identify multiple overlapping images, apart from using the sliding window technique, we need to use bounding boxes for each of the object in sliding window. So, we will add another layer in between the sliding window layer and max pooling layer that predicts the potential bounding boxes that contain parts of objects(because objects are occluded) in the output of the sliding window layer. The boxes can be predicted by using some heuristics like for example,
We have some number of boxes of different sizes and we predict that a box is the bounding for an object if the center of the object lies in the box. If multiple boxes are mapped to same object, we can use that box, that has maximum overlapping area or combine the box boundaries.

Then we apply maxpooling and further layers on these boxes to predict objects in them. In the final layer, we can use softmax to determine the probability of the object in the box to belong to a certain class, but, predict that class only if the probability is above a certain threshold. Finally, after the final fully connected softmax layer, we can use a linear regressor to fit the boxes to the objects, assuming we know the positions of the objects in the images in the training data and their bounding boxes. This will help the network to recognise the boxes for different objects in an image like cars, bikes etc..
Limitation: This method can correctly identify objects if there is atleast some minimum visibility of atleast some features of the object to confidently predict the class of the object. If any object is very much occluded by other objects, then it may not be possible to classify it. Also, it’s computationally expensive mainly because of the layer that identifies boxes to be put around objects in images, so it can’t be used in real-time.