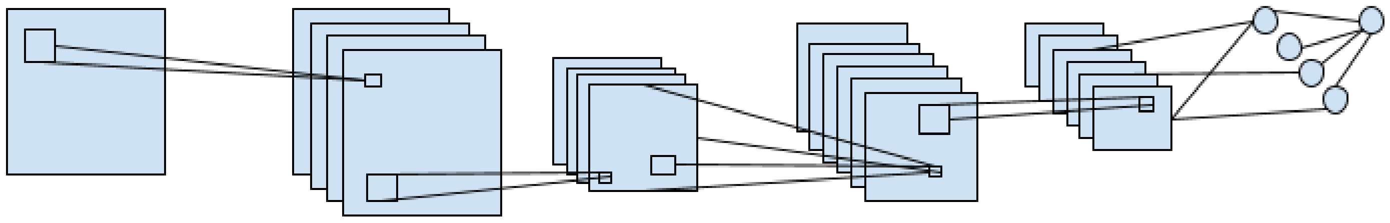
Homework 5

Question 1)

1) The Gaussian Mixture Model (GMM) is a method that assumes data is generated by a mix of different Gaussian distributions. The parameters in the distributions are unknown. Each Gaussian distribution, or component, represents a cluster in the data shaped like an ellipsoid. A GMM consists of k Gaussian distributions, where k is the number of clusters, a value that needs to be known beforehand. In this model, we assume that the dataset X is created through a probabilistic process. For each instance, a cluster is randomly chosen from the k clusters. The probability of selecting a cluster is determined by the cluster's weight, often determined using the maximum likelihood solution. The algorithm employed for this is the expectation-maximization (EM) algorithm, which simplifies complex problems by introducing hidden variables. In the context of a GMM with two Gaussian components, the EM algorithm seeks to find the maximum likelihood solution. It does this by introducing a latent variable f to represent the unknown Gaussian component from which each data point originates. The EM algorithm iteratively performs an E-step, where it computes the expectation of this variable based on the data, and an M-step, where it maximizes this expectation over the model parameters, including means, standard deviations, and mixing parameters. These steps are repeated until the algorithm converges, reaching a local maximum. Gaussian mixture models prove valuable in anomaly detection through the establishment of a density threshold. Instances situated in regions of low density are identified as anomalies. Adjusting the density threshold allows for fine-tuning the balance between false positives and false negatives. Lowering the threshold addresses an excess of false positives, while raising it mitigates the occurrence of false negatives. This adaptability enables effective customization of the model's sensitivity to anomalies based on specific application requirements.

2)

1 2 3 4 5
6



I have labeled each of the layers above with numbers. Numbers correspond to the layers below:

1: Input layer - represents the raw input data, usually an image or a set of images. Each input channel corresponds to a different feature (i.e. red, green, and blue channels in a color image).

2 and 4: Convolution layer - The convolution layer is used to extract the various features from the image input. Convolution is performed on the input using a filter or kernels. The filters slide over the input to detect patterns, edges, and features. Multiple filters create multiple feature maps, capturing different aspects of the input. Number 2’s convolution layer consists of 4 feature maps and number 4’s convolution layer consists of 6 feature maps.

3 and 5: Pooling/sub-sampling layers - this layer has a pool size and divides each spatial dimension by a factor of that size. Each neuron in a pooling layer is connected to the outputs of a limited number of neurons in the previous layer. It is a method to reduce the size of the convolved

feature map. Pooling helps reduce the computational load and makes the network more robust to variations in scale and orientation. Pooling also introduces invariance to small translations. Types of pooling include max pooling and average pooling. In average pooling. Number 3's pooling layer consists of 4 feature maps and number 5's pooling layer contains 6 feature maps.

6: Fully connected MLP - These layers connect every neuron in one layer to every neuron in the next layer. They contain weights and biases along with the neurons. Fully connected layers at the end of the network are responsible for making predictions or classifying the input based on the learned features.

The final layer outputs the prediction.

Several state-of-the-art CNN architectures have emerged, each with distinct characteristics and advancements in deep learning. One such architecture is LeNet-5, featuring seven layers comprising three convolutional layers, two subsampling layers, and two fully connected layers. Another noteworthy architecture is AlexNet, a larger and deeper variant of LeNet-5. AlexNet pioneered the stacking of convolutional layers directly on top of one another.
GoogLeNet represents a significant stride in depth, surpassing its predecessors by incorporating inception modules for enhanced efficiency. The architecture landscape also includes VGG-16, Inception-v1, Inception-v3, ResNet-50, Xception, Inception-v4, Inception ResNets, and ResNeXt-50, each contributing to the evolving landscape of convolutional neural networks.

3) The vanishing gradients problem emerges when gradients diminish progressively during the backpropagation algorithm's descent through lower layers. Consequently, Gradient Descent updates leave the connection weights of these lower layers nearly unchanged, preventing the training process from converging to an optimal

solution. On the contrary, when gradients amplify exponentially, resulting in excessively large weight updates and algorithmic divergence, it is referred to as exploding gradients. Addressing these challenges, one effective technique is to improve the weight initialization. This can be accomplished by using Glorot initialization, LeCun initialization, and He initialization. This method ensures that the variance of outputs for each layer equals the variance of its inputs, and that gradients maintain equal variance both before and after traversing a layer in the reverse direction. Essentially, this proposed solution involves initializing the connection weights of each layer randomly.

Another strategy involves utilizing nonsaturating activation functions, such as the leaky ReLU function. By setting the slope of negative input to a small value, this variant ensures that leaky ReLUs never become inactive; they have the potential to wake up over time. There are other variants of this such as Randomized Leaky ReLU (RReLU) and parametric leaky ReLU (PReLU) Additionally, batch normalization is employed, introducing an operation before or after the activation function of each hidden layer. This operation zero-centers and normalizes each input, subsequently scaling and shifting the result using two new parameter vectors per layer. The algorithm accomplishes this by estimating mean and standard deviation for each input, based on the current mini-batch. Through this method, the vanishing gradient problem is significantly reduced.

Additionally, gradient clipping serves as a protective measure against exploding gradients. This technique involves constraining gradients during backpropagation, preventing them from surpassing a predefined threshold.

Question 2

When h is tested on a set of 100 examples, it classifies 80 correctly, so it classifies 100-80 = 20 incorrectly.

$$error_S(h) = \frac{20}{100} = 0.20$$

We know that $z_{95} = 1.96$ $n = 100$

So the 95% confidence interval for the true error rate for $Error_D(h)$ is

$$0.20 \pm 1.96 \sqrt{\frac{0.20(1-0.20)}{100}}$$

or $0.20 \pm 1.96 \cdot 0.04$

$= 0.20 \pm 0.0784$

$= [0.1216, 0.2784]$