1. What are the advantages of a CNN for image classification over a completely linked DNN?

Ans. Convolution neural networks are mainly credited for their role in image processing. before CNN, manual labor was involved in identifying objects in images. CNN helped scale the process by using linear algebra principles to identify patterns in images.

The working of CNNs is based on the three main layers. These are:

Convolutional layer

Pooling layer

Fully-connected layer

With each layer, the CNN's complexity in understanding the image increases. This means that layers at the beginning are responsible for detecting low-level features such as edges and colors and the layers at the end are responsible for detecting high-level features such as shapes that we can easily recognize.

The main advantage of CNNs compared to a traditional neural network is that they automatically detect important features without any human supervision. For example, given any pictures of cats and dogs, it can learn the key features for each class by itself. Besides, the number of parameters learned during training reduces significantly since all the features have been extracted already.

With ANN, concrete data points must be provided. For example, in a model where we are trying to distinguish between dogs and cats, the width of the noses and length of the ears must be explicitly provided as data points. This is increases the number of parameters to be learned during training.

2. Consider a CNN with three convolutional layers, each of which has three kernels, a stride of two, and SAME padding. The bottom layer generates 100 function maps, the middle layer 200, and the top layer 400. RGB images with a size of 200 x 300 pixels are used as input. How many criteria does the CNN have in total? How much RAM would this network need when making a single instance prediction if we're using 32-bit floats? What if you were to practice on a batch of 50 images?

Ans. parameters

first convolutional layer kernel-size and RGB channels, plus bias: 3 * 3 * 3 + 1 = 28 output feature maps is 100: 28 * 100 = 2800

second convolutional layer kernel-size and last feature maps, plus bias: 3 * 3 * 100 + 1 = 901 output feature maps is 200: 901 * 200 = 180200

third convolutional layers kernel-size and last feautre maps, plus bias: 3 * 3 * 200 + 1 =1801 output feautre maps is 400: 1801 * 400 = 720400

Total parameters is 2800 + 180200 + 720400 = 903400

memories since 32-bit is 4 bytes

first convolutional layer one feature map size: 100 * 150 = 15000 total output: 15000 * 100 = 1,500,000

second convolutional layer one feature map size: 50 * 75 = 3,750 total output: 3750 * 200 = 750,000

third convolutional layer one feature map size: 25 * 38 = 950 total ouput: 950 * 400 = 380, 000

(1,500,000 + 750,000 + 380,000) * 4 / 1024 /1024 = 10.032 (MB) 903400 * 4 / 1024 / 1024 = 3.44 (MB) 10.032+ 3.44=13.47(MB)

3. What are five things you might do to fix the problem if your GPU runs out of memory while

training a CNN?

1. reduce mini-batch size
2. reduce dimensionality using a larger stride in one or more layers
3. remove one or more layers
4. using 16-bits instead of 32-bit floats
5. distributed the cnn across multiple devices

4. Why would you use a max pooling layer instead with a convolutional layer of the same stride?

Ans. Max-pooling helps in extracting low-level features like edges, points, etc. While Avg-pooling goes for smooth features. If time constraint is not a problem, then one can skip the pooling layer and use a convolutional layer to do the same.

5. When would a local response normalization layer be useful?

Ans. Normalization is a pre-processing technique used to standardize data. In other words, having different sources of data inside the same range. Not normalizing the data before training can cause problems in our network, making it drastically harder to train and decrease its learning speed.

6. In comparison to LeNet-5, what are the main innovations in AlexNet? What about GoogLeNet and

ResNet's core innovations?

- AlexNet
    - it is much larger and deeper

    - stacks convolutional layer directly on top of each convlutional layer
- GooLeNet
    - introduce a *inception modules*, which make it possible to have much deeper net than previous network
- ResNet
    - introduce a skip connection.

7. On MNIST, build your own CNN and strive to achieve the best possible accuracy.

8. Using Inception v3 to classify broad images. a.

Images of different animals can be downloaded. Load them in Python using the matplotlib.image.mpimg.imread() or scipy.misc.imread() functions, for example. Resize and/or crop them to 299 x 299 pixels, and make sure they only have three channels (RGB) and no transparency. The photos used to train the Inception model were preprocessed to have values ranging from -1.0 to 1.0, so make sure yours do as well.

9. Large-scale image recognition using transfer learning.

a. Make a training set of at least 100 images for each class. You might, for example, identify your own photos based on their position (beach, mountain, area, etc.) or use an existing dataset, such as the flowers dataset or MIT's places dataset (requires registration, and it is huge).

b. Create a preprocessing phase that resizes and crops the image to 299 x 299 pixels while also adding some randomness for data augmentation.

c. Using the previously trained Inception v3 model, freeze all layers up to the bottleneck layer (the last layer before output layer) and replace output layer with appropriate number of outputs for your new classification task (e.g., the flowers dataset has five mutually exclusive classes so the output layer must have five neurons and use softmax activation function).

d. Separate the data into two sets: a training and a test set. The training set is used to train the model, and the test set is used to evaluate it.