1. What are the advantages of a CNN over a fully connected DNN for image classification?

Ans. Convolutional Neural Networks (CNN) are everywhere. It is arguably the most popular deep learning architecture. The recent surge of interest in deep learning is due to the immense popularity and effectiveness of convnets. The interest in CNN started with AlexNet in 2012 and it has grown exponentially ever since. In just three years, researchers progressed from 8 layer AlexNet to 152 layer ResNet.

The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. For example, given many pictures of cats and dogs it learns distinctive features for each class by itself.CNN is also computationally efficient. It uses special convolution and pooling operations and performs parameter sharing. This enables CNN models to run on any device, making them universally attractive.All in all this sounds like pure magic. We are dealing with a very powerful and efficient model which performs automatic feature extraction to achieve superhuman accuracy.

The main building block of CNN is the convolutional layer. Convolution is a mathematical operation to merge two sets of information. In our case the convolution is applied on the input data using a convolution filter to produce a feature map. There are a lot of terms being used so let’s visualize them one by one.

1. Consider a CNN composed of three convolutional layers, each with 3 × 3 kernels, a stride of 2, and "same" padding. The lowest layer outputs 100 feature maps, the middle one outputs 200, and the top one outputs 400. The input images are RGB images of 200 × 300 pixels.

Ans. 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

What is the total number of parameters in the CNN? If we are using 32-bit floats, at least how much RAM will this network require when making a prediction for a single instance? What about when training on a mini-batch of 50 images?

Ans. The first convolutional layer has

Kernels = 3 x 3

Inputs Channels = 3

No of filters = 1

So each feature map has (3 x 3 x 3) x 1 = 27 weights + 1 bias = 28 weights

So, for 100 feature maps, we have 28 \* 100 = 2800 parameters

The second convolutional layer has

Input = 100 (i.e feature maps from previous layer)

kernel = 3 x 3

total weights = 3 \* 3 \* 100 = 900 + 1 (bias) = 901

So, 1 feature map has 901 weights, then 200 feature maps has 901 \* 200 = 180200 parameters

Similarly for third Convolutional layer has 720,400 parameters

The total number of parameters are 720,400 + 180200 + 2800 = 903,400 parameters

1. If your GPU runs out of memory while training a CNN, what are five things you could try to solve the problem?

Ans. 1) first of all you can see how much memory it gets when it runs by monitoring your gpu. for example if you have a nvidia gpu u can check that with watch -n 1 nvidia-smi command. But in most cases if you didn't set the maximum fraction of gpu memory, it allocates almost the whole free memory. your problem is lack of enough memory for your gpu. cnn networks are totally heavy. When you are trying to feed your network DO NOT do it with your whole data. DO this feeding procedure in low batch sizes.

2) If you encountering out of memory errors on a GTX 970 then there are 2 ways to solve :

The first is the allow\_growth option, which attempts to allocate only as much GPU memory based on runtime allocations:

config = tf.ConfigProto() config.gpu\_options.allow\_growth = True session = tf.Session(config=config)

The second method is the per\_process\_gpu\_memory\_fraction option, which determines the fraction of the overall amount of memory that each visible GPU should be allocated. For example, you can tell TensorFlow to only allocate 40% of the total memory of each GPU by:

config = tf.ConfigProto() config.gpu\_options.per\_process\_gpu\_memory\_fraction = 0.4 session = tf.Session(config=config)

1. Why would you want to add a max pooling layer rather than a convolutional layer with the same stride?

Ans. The functions of pooling layers and convolution layers are different in a CNN.

Convolution layers are used to match identical features between different images. For example, if we are looking for edge like features we would like to take the filter that resembles an edge feature and super impose it with the image. If we increase the stride, we are skipping certain parts of the image. Which means we are ignoring parts of the image that might have an edge feature. Therefore this is not advised as method to reduce the size of image.

Pooling layers are used to reduce the size of the image as having a very big image can increase computational complexity and sometimes induce noise.

Therefore, the purpose of both are different and hence the one can’t be supplanted by the other.Also A Pooling layer is totally different from a convolutional layer.A convolutional layer consists of independent filters and each filter is convolved (combined) with the input image.A convolutional layer with 6 filters having size of 5x5x3 is moved over an image and the dot product is taken between them.These feature maps obtained from convolutional layers are used for classification , but using all the extracted features is computationally expensive.So one approach is to aggregate statistics of these features at various locations like calculating the max , min or mean value of a region over an image. This operation is called the Pooling operation.So a Pooling layer is not necessary but advisable to reduce the number of extracted features and to avoid overfitting.

1. When would you want to add a local response normalization layer?

Ans. So basically Local Response Normalization (LRN) layer implements the lateral inhibition we were talking about in the previous section. This layer is useful when we are dealing with ReLU neurons. Why is that? Because ReLU neurons have unbounded activations and we need LRN to normalize that. We want to detect high frequency features with a large response. If we normalize around the local neighborhood of the excited neuron, it becomes even more sensitive as compared to its neighbors.

At the same time, it will dampen the responses that are uniformly large in any given local neighborhood. If all the values are large, then normalizing those values will diminish all of them. So basically we want to encourage some kind of inhibition and boost the neurons with relatively larger activations.

We want to add it because A typical CNN consists of the following layers: convolution, pooling, rectified linear unit (ReLU), fully connected, and loss. If the previous sentence didn’t make sense, you may want to go through a quick CNN tutorial before proceeding further. Anyway, the reason we may want to have normalization layers in our CNN is that we want to have some kind of inhibition scheme.

In neurobiology, there is a concept called “lateral inhibition”. Now what does that mean? This refers to the capacity of an excited neuron to subdue its neighbors. We basically want a significant peak so that we have a form of local maxima. This tends to create a contrast in that area, hence increasing the sensory perception. Increasing the sensory perception is a good thing! We want to have the same thing in our CNNs.

1. Can you name the main innovations in AlexNet, compared to LeNet-5? What about the main innovations in GoogLeNet, ResNet, SENet, and Xception?

Ans. The main innovations over LeNet-5 are:

The search space is much larger and deeper

it has convolutional layers directly on top of them, instead of pooling layers on the top of each convolutional layer.

The main innovation in GoogLeNet is the introduction of inception modules, which helps in having much deeper net than previous CNN architectures, with lesser parameters.

ResNet’s main innovation is the introduction of alternative connections, which make it possible to go well beyond cent layers.

The easiness and consistency in ResNet's are also rather appreciative and innovative.

VGGNet is the upgradation over the AlexNet , It helps in reducing the training time and hence making the system more faster.

1. What is a fully convolutional network? How can you convert a dense layer into a convolutional layer?

Ans. Ans: FCN is a network that does not contain any “Dense” layers (as in traditional CNNs) instead it contains 1x1 convolutions that perform the task of fully connected layers (Dense layers).

(FCN) uses a convolutional neural network to transform image pixels to pixel category convolutional neural networks previously introduced, an FCN transforms the height and width of the intermediate layer feature map back to the size of input image through the transposed convolution layer, so that the predictions have a one-to-one correspondence with input image in spatial dimension (height and width). Given a position on the spatial dimension, the output of the channel dimension will be a category prediction of the pixel corresponding to the location.

A fully convolution network can be built by simply replacing the FC layers with there equivalent Conv layers. In the example of VGG16 we can do so by first removing the last four layers. One way to do so is to pop layers from the model. In the model stack, each popping will remove the last layer.

One benefit of replacing a fully connected layer with a convolutional layer is that the number of parameters to adjust are reduced due to the fact that the weights are shared in a convolutional layer.

This means faster and more robust learning. Additionally max pooling can be used just after a convolutional layer to reduce the dimensionality of the layer.

This means improved robustness to distortions in input stimuli and a better overall performance.

Four decades back, neural networks were only two layers deep as it was not computationally feasible to build larger networks. Now, it is common to have neural networks with 10+ layers and even 100+ layer ANNs are being tried upon

Deep Learning Applications

1. Virtual Assistants. Virtual Assistants are cloud-based applications that understand natural language voice commands and complete tasks for the user.

2. Chatbots. Chatbots can solve customer problems in seconds.

3. Healthcare.

4. Entertainment.

5. News Aggregation and Fake News Detection.

6. Composing Music.

7. Image Colouring.

8. Robotics.

Deep learning is a sub-branch of AI and ML that follow the workings of the human brain for processing the datasets and making efficient decision making.

Practical examples of deep learning are Virtual assistants, vision for driverless cars, money laundering, face recognition and many more.

1. What is the main technical difficulty of semantic segmentation?
2. Build your own CNN from scratch and try to achieve the highest possible accuracy on MNIST.
3. Use transfer learning for large image classification, going through these steps:
   1. Create a training set containing at least 100 images per class. For example, you could classify your own pictures based on the location (beach, mountain, city, etc.), or alternatively you can use an existing dataset (e.g., from TensorFlow Datasets).
   2. Split it into a training set, a validation set, and a test set.
   3. Build the input pipeline, including the appropriate preprocessing operations, and optionally add data augmentation.
   4. Fine-tune a pretrained model on this dataset.