Q1: What are the advantages of a CNN over a fully connected DNN for image classification?

Ans: CNNs have several advantages over fully connected DNNs for image classification, including the ability to learn local features and patterns, translation invariance, and fewer parameters, which can lead to better generalization and reduced overfitting.

Q2: 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. 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 total number of parameters in the CNN is 1,199,600. When making a prediction for a single instance using 32-bit floats, this network will require at least 90.4 MB of RAM. When training on a mini-batch of 50 images, this network will require at least 4.52 GB of RAM.

Q3: If your GPU runs out of memory while training a CNN, what are five things you could try to solve the problem?

Ans: Some potential solutions to a GPU running out of memory while training a CNN include reducing the batch size, reducing the image size or resolution, reducing the number of layers or parameters in the network, using a lower-precision data type (e.g., 16-bit floats), or using gradient checkpointing.

Q4: Why would you want to add a max pooling layer rather than a convolutional layer with the same stride?

Ans: Max pooling layers are often used instead of convolutional layers with the same stride because they can reduce the spatial dimensions of the feature maps while preserving important features, making the network more efficient and reducing overfitting.

Q5: When would you want to add a local response normalization layer?

Ans: Local response normalization layers were commonly used in early CNN architectures to enhance the contrast between features and reduce the effect of "competition" between neurons. However, they are less commonly used now because other techniques, such as batch normalization, have been shown to be more effective.

Q6: 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 in AlexNet, compared to LeNet-5, include the use of a deeper network with more layers and more feature maps, the use of ReLU activation functions, and the use of dropout to reduce overfitting. The main innovations in GoogLeNet include the use of inception modules with multiple different kernel sizes, the use of global average pooling, and the use of auxiliary classifiers. The main innovation in ResNet is the use of residual connections, which allow for the training of very deep networks by preventing the vanishing gradient problem. The main innovation in SENet is the use of squeeze-and-excitation blocks, which selectively weight the feature maps based on their importance. The main innovation in Xception is the use of depthwise separable convolutions, which separate the spatial and channel-wise convolutions in a network to make it more efficient and reduce the number of parameters.

Q7: What is a fully convolutional network? How can you convert a dense layer into a convolutional layer?

Ans: A fully convolutional network (FCN) is a type of neural network that uses only convolution layers for both feature extraction and classification, rather than fully connected layers. FCNs are commonly used for image segmentation and other tasks where the output is a spatially dense prediction.

To convert a dense layer into a convolutional layer, the weights of the dense layer can be reshaped into a 4D tensor, where the first two dimensions correspond to the filter size, the third dimension corresponds to the number of input channels, and the fourth dimension corresponds to the number of output channels. The resulting tensor can then be used as the weights of a convolutional layer. This allows the dense layer to be used in a convolutional network and take advantage of its benefits, such as weight sharing and spatial invariance.