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

ANS:

CNNs do not require human supervision for the task of identifying important features. They are very accurate at image recognition and classification. Weight sharing is another major advantage of CNNs. Convolutional neural networks also minimize computation in comparison with a regular neural network.

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.

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:

A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

Example: Input volumn of 32x32x3, what is the output size if we're to apply 10 5x5 filters with stride 1 and pad 2. And what is the total number of weights for this layer? The output width and the height will be ( 32 - 5 + 2 \* 2 ) / 1 + 1 = 32 and the depth will be 10.

The number of neurons for all layers after the first is clear. One simple way to calculate the neurons is to simply multiply the three dimensions of that layer ( planes X width X height ): Layer 2: 27x27x128 \* 2 = 186,624. Layer 3: 13x13x192 \* 2 = 64,896.

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

ANS:

This error message indicates that a project is too complex to be cached in the GPU's memory. Each project contains a certain amount of data that must be uploaded either to the RAM (when the CPU engine is chosen) or to the VRAM (the GPU's memory, when the GPU engine is CUDA or RTX).

Running more apps depletes the amount of RAM available on your PC. Closing superfluous apps and other background processes may free up enough RAM on your PC to at least temporarily resolve the “low on memory” error. Even the message for that error suggests you close active programs.

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

ANS:

It was found that applying the pooling layer after the convolution layer improves performance helping the network to generalize better and reduce overfitting.

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

ANS:

This is useful when we are dealing with ReLU neurons. ReLU neurons have unbounded activations, and we need local response normalization (LRN) to normalize them. To do this, we need to identify high frequency features. By applying LRN, the neurons becomes more sensitive than their neighbors.

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 innovation introduced by AlexNet compared to the LeNet-5 was its sheer size. AlexNet main elements are the same: a sequence of convolutional and pooling layers followed by a couple of fully-connected layers

What about the main innovations in GoogLeNet and ResNet? The main innovations in AlexNet compared to LeNet-5 are (1) it is much larger and deeper, and (2) it stacks convolutional layers directly on top of each other, instead of stacking a pooling layer on top of each convolutional layer.

AlexNet has parallel two CNN line trained on two GPUs with cross-connections, GoogleNet has inception modules ,ResNet has residual connections.

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

ANS:

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.

1. What is the main technical difficulty of semantic segmentation?

ANS:

The problems of semantic segmentation and the way of solving these problems. The over-reliance on accurate labeling makes the existing semantic segmentation methods difficult to be directly applied to imprecise or even unlabeled datasets.

1. Build your own CNN from scratch and try to achieve the highest possible accuracy on MNIST.

Ans:

The third model uses both of those techniques plus batch normalization. The last model employs all three of those techniques plus data augmentation and achieves the best accuracy of 99.5%!

LeNet is the first successful application of CNNs and was developed by Yann Lecun in the 1990s that was used to read zip codes, digits, etc. The latest work is called LeNet-5 which a 5-layer CNN that reaches 99.2 % accuracy on insolated character recognition.

1. 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).

ANS:

dataset, info = tfds.load("food101",

shuffle\_files=False,

with\_info=True,

as\_supervised=True)

Downloading and preparing dataset 4.65 GiB (download: 4.65 GiB, generated: Unknown size, total: 4.65 GiB) to /root/tensorflow\_datasets/food101/2.0.0...

Dataset food101 downloaded and prepared to /root/tensorflow\_datasets/food101/2.0.0. Subsequent calls will reuse this data.

2022-05-22 01:42:56.525982: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-05-22 01:42:56.648735: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-05-22 01:42:56.649537: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-05-22 01:42:56.653453: I tensorflow/core/platform/cpu\_feature\_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2 AVX512F FMA

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

2022-05-22 01:42:56.655431: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-05-22 01:42:56.656437: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-05-22 01:42:56.657378: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-05-22 01:42:58.789808: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-05-22 01:42:58.790612: I tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-05-22 01:42:58.791290: I tensorflow/stream\_executor/cuda/cuda\_gpu\_/\*.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

2022-05-22 01:42:58.793145: I tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1510] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 15403 MB memory: -> device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0

class\_names = info.features["label"].names

class\_names

Out[7]:

['apple\_pie',

'baby\_back\_ribs',

'baklava',

'beef\_carpaccio',

'beef\_tartare',

'beet\_salad',

'beignets',

'bibimbap',

'bread\_pudding',

'breakfast\_burrito',

'bruschetta',

'caesar\_salad',

'cannoli',

'caprese\_salad',

'carrot\_cake',

'ceviche',

'cheesecake',

'cheese\_plate',

'chicken\_curry',

'chicken\_quesadilla',

'chicken\_wings',

'chocolate\_cake',

'chocolate\_mousse',

'churros',

'clam\_chowder',

'club\_sandwich',

'crab\_cakes',

'creme\_brulee',

'croque\_madame',

'cup\_cakes',

'deviled\_eggs',

'donuts',

'dumplings',

'edamame',

'eggs\_benedict',

'escargots',

'falafel',

'filet\_mignon',

'fish\_and\_chips',

'foie\_gras',

'french\_fries',

'french\_onion\_soup',

'french\_toast',

'fried\_calamari',

'fried\_rice',

'frozen\_yogurt',

'garlic\_bread',

'gnocchi',

'greek\_salad',

'grilled\_cheese\_sandwich',

'grilled\_salmon',

'guacamole',

'gyoza',

'hamburger',

'hot\_and\_sour\_soup',

'hot\_dog',

'huevos\_rancheros',

'hummus',

'ice\_cream',

'lasagna',

'lobster\_bisque',

'lobster\_roll\_sandwich',

'macaroni\_and\_cheese',

'macarons',

'miso\_soup',

'mussels',

'nachos',

'omelette',

'onion\_rings',

'oysters',

'pad\_thai',

'paella',

'pancakes',

'panna\_cotta',

'peking\_duck',

'pho',

'pizza',

'pork\_chop',

'poutine',

'prime\_rib',

'pulled\_pork\_sandwich',

'ramen',

'ravioli',

'red\_velvet\_cake',

'risotto',

'samosa',

'sashimi',

'scallops',

'seaweed\_salad',

'shrimp\_and\_grits',

'spaghetti\_bolognese',

'spaghetti\_carbonara',

'spring\_rolls',

'steak',

'strawberry\_shortcake',

'sushi',

'tacos',

'takoyaki',

'tiramisu',

'tuna\_tartare',

'waffles']

In [8]:

linkcode

n\_classes = info.features["label"].num\_classes

n\_classes

101

In [9]:

dataset\_size = info.splits["train"].num\_examples

dataset\_size

Out[9]:

75750

* 1. Split it into a training set, a validation set, and a test set.

ANS:

In general, putting 80% of the data in the training set, 10% in the validation set, and 10% in the test set is a good split to start with. The optimum split of the test, validation, and train set depends upon factors such as the use case, the structure of the model, dimension of the data, etc.

* 1. Build the input pipeline, including the appropriate preprocessing operations, and optionally add data augmentation.

ANS:

The input pipeline is a quick and easy utility provided in tf. data api to make complex input pipelines from simple and reusable codes and all in few lines of code. It also allows handling a large amount of data, thus giving low-end machines an advantage in computing them. It does it by wrapping the data into tf

* 1. Fine-tune a pretrained model on this dataset.

ANS:

When the model is trained on a large generic corpus, it is called 'pre-training'. When it is adapted to a particular task or dataset it is called as 'fine-tuning'. Technically speaking, in either cases ('pre-training' or 'fine-tuning'), there are updates to the model weights.