1. What are the advantages of a CNN over a fully connected DNN for image classification?
2. 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?

1. If your GPU runs out of memory while training a CNN, what are five things you could try to solve the problem?
2. Why would you want to add a max pooling layer rather than a convolutional layer with the same stride?
3. When would you want to add a local response normalization layer?
4. Can you name the main innovations in AlexNet, compared to LeNet-5? What about the main innovations in GoogLeNet, ResNet, SENet, and Xception?
5. What is a fully convolutional network? How can you convert a dense layer into a convolutional layer?
6. What is the main technical difficulty of semantic segmentation?
7. Build your own CNN from scratch and try to achieve the highest possible accuracy on MNIST.
8. 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.