Questions

"Training"

1. What are the advantages of convolutional layers over fully-connected layers?
2. PyTorch convolutions expect data in the "NCHW" format. What does this mean?
3. Why does it make sense to share weights in a convolutional neural network?
4. What is the purpose of zero padding in a convolutional layer?
5. Why should we design our neural networks so that the number of activations in each layer generally **decreases**?
6. Explain the idea of transfer learning, specifically of using AlexNet features to train a neural network more quickly.
7. What is the output of the following code?

```
>>> x = torch.randn(20, 3, 16, 16) # NCHW
>>> conv = nn.Conv2d(in_channels=3, out_channels=7, kernel_size=5, padding=0)
>>> conv(x).shape
```

8. What new idea was introduced in GoogLeNet? ResNet?
9. What are fully convolutional networks? Name one advantage and one disadvantage to using a fully convolutional network.

"Generalization"

1. How many multiplications are performed when applying a 3x3 convolution on a greyscale image of size 7x7?
2. Which operations to reduce the number of units in the hidden layer are the *most* and *least* computationally efficient: max pooling, average pooling, and using strides in a convolution?
3. Suppose we learn a fully-connected network on a 128x128 image. Can we use the same network to make predictions about images of size 256x256? Why or why not?
4. Suppose we learn a fully convolutional network on a 128x128 image. Can we use the same network to make predictions about images of size 256x256? Why or why not?