Introduction

In this section of the course you will be building a memory-efficient CNN known as SqueezeNet. Although small, its performance is on par with much larger models such as AlexNet, the breakthrough deep CNN that won the 2012 ImageNet Challenge. We will train the model on the CIFAR-10 dataset, which contains 60,000 images in 10 different classes.

A. Memory Usage

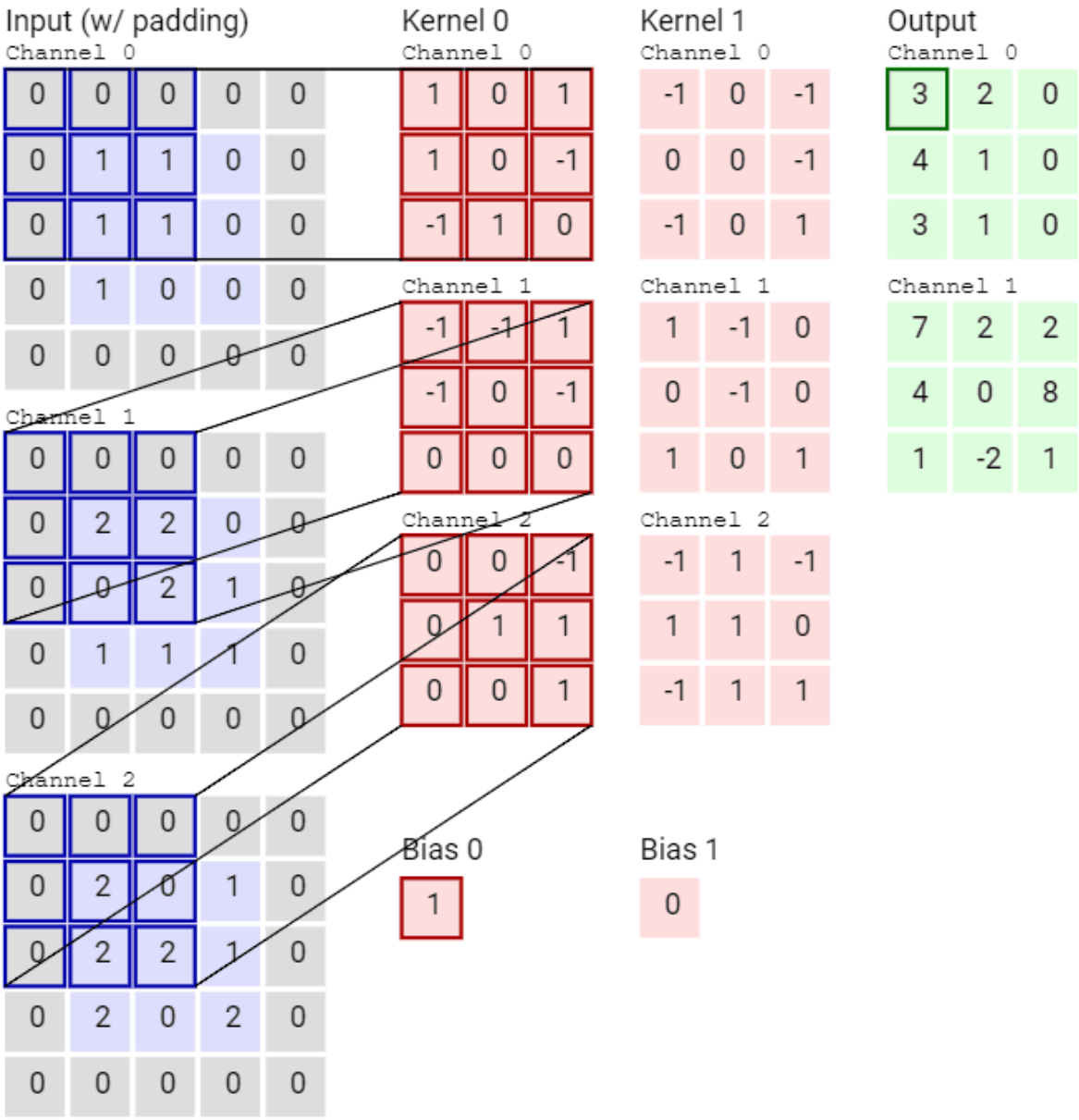
While we're normally concerned with model accuracy, the amount of memory a model uses is important as well. After training a model, we store its computation graph and parameters (weights + biases) for future use. Though a model's computation graph is relatively small (since even large models won't have more than a couple hundred layers), the number of parameters a model has can be in the millions.

Most high performance models require hundreds of MB of space to store their parameters. The aforementioned AlexNet uses over 200MB for storage of 60 million parameters. On the other hand, the SqueezeNet model architecture uses less than 1MB. That's even less memory than the simple digit recognition model we built in the previous chapter (13MB). However, SqueezeNet has significantly better performance than our previous model and actually matches AlexNet in accuracy.

When accuracy between two models is relatively equal, we prefer the model that uses up less memory.

B. Calculating parameters

To understand model sizes, we need to be able to calculate the number of parameters in a model. Let’s take a look at an example convolution layer:



Convolution layer with 2 filters. The input data has 3 channels and each kernel has dimension 3x3.

The convolution layer parameters are the kernel weights and biases. Each kernel has 3x3 dimensions, and there are 3 kernels for each of the 2 filters. Therefore, the total number of kernel weights is $3 \times 3 \times 2 \times 3 = 54$. There are 2 biases (one per filter), so the total number of parameters in this convolution layer is 56. In general, for a convolution layer with C input channels, F filters, and kernels with dimension $H_K \times W_K$, the number of parameters, P, is

$$P = H_K \times W_K \times F \times C + F$$

To calculate the total number of parameters for a model, we just aggregate the number of parameters across each convolution layer.

C. SqueezeNet

There are quite a few benefits of using a smaller model such as SqueezeNet. First, it takes less time to train and load a smaller model. When performance is comparable, it is preferable to use a model that takes less time to train and set up.

Furthermore, when memory usage is limited, it may not even be possible to store a large model. For example, if you're working in an environment with a limited disk quota (such as a Linux container), storing a large model may exceed the memory limit. This problem is especially prevalent when we are storing multiple variations of a large model (which is often the case during training and experimentation). However, this would not be an issue with a model based on the SqueezeNet architecture.