

says that the function the layer should learn contains only local interactions and is equivariant to translation. Likewise, the use of pooling is an infinitely strong prior that each unit should be invariant to small translations.

Of course, implementing a convolutional net as a fully connected net with an infinitely strong prior would be extremely computationally wasteful. But thinking of a convolutional net as a fully connected net with an infinitely strong prior can give us some insights into how convolutional nets work.

One key insight is that convolution and pooling can cause underfitting. Like any prior, convolution and pooling are only useful when the assumptions made by the prior are reasonably accurate. If a task relies on preserving precise spatial information, then using pooling on all features can increase the training error. Some convolutional network architectures (Szegedy *et al.*, 2014a) are designed to use pooling on some channels but not on other channels, in order to get both highly invariant features and features that will not underfit when the translation invariance prior is incorrect. When a task involves incorporating information from very distant locations in the input, then the prior imposed by convolution may be inappropriate.

Another key insight from this view is that we should only compare convolutional models to other convolutional models in benchmarks of statistical learning performance. Models that do not use convolution would be able to learn even if we permuted all of the pixels in the image. For many image datasets, there are separate benchmarks for models that are **permutation invariant** and must discover the concept of topology via learning, and models that have the knowledge of spatial relationships hard-coded into them by their designer.

## 9.5 Variants of the Basic Convolution Function

When discussing convolution in the context of neural networks, we usually do not refer exactly to the standard discrete convolution operation as it is usually understood in the mathematical literature. The functions used in practice differ slightly. Here we describe these differences in detail, and highlight some useful properties of the functions used in neural networks.

First, when we refer to convolution in the context of neural networks, we usually actually mean an operation that consists of many applications of convolution in parallel. This is because convolution with a single kernel can only extract one kind of feature, albeit at many spatial locations. Usually we want each layer of our network to extract many kinds of features, at many locations.