Convnets

Prof. Eric A. Suess

April 13, 2020
Chapter 5 covers

- Understanding **convolutional neural networks** (convnets)
- Using **data augmentation** to mitigate overfitting
- Using a **pretrained convnet** to do feature extraction
- Fine-tuning a pretrained convnet
- **Visualizing** what convnets learn and how they make classification decisions
Convolutional neural networks revolutionized the field of computer vision and are now the main type of deep learning neural networks that are used in this field.

**Spotlight Blog:** Medium How to easily Detect Objects with Deep Learning on Raspberry Pi

kaggle ImageNet

ImageNet
MNIST again

- Earlier, using a feed forward neural network, the accuracy was 97.8%.
- Now, using a convolutional neural network, the accuracy can be improved, beyond 99%.
Convnets work locally

What is the difference between a densely connected layer and a convolution layer?

- Dense layers learn global patterns in their input feature space
- Convolution layers learn local patterns
Convnets have an interesting property

- The patterns they learn are translation invariant.
- They can learn spatial hierarchies of patterns.
Convnets

- Convolutions operate over 3D tensors, called **feature maps**, with two spatial axes (height and width) as well as a depth axis (also called the channels axis).
- For an **RGB** image, the dimension of the depth axis is 3, because the image has three color channels: red, green, and blue.
- For a **black-and-white** picture, like the MNIST digits, the depth is 1 (levels of gray).
“In the MNIST example, the first convolution layer takes a feature map of size (28, 28, 1) and outputs a feature map of size (26, 26, 32): it computes 32 filters over its input. Each of these 32 output channels contains a $26 \times 26$ grid of values, which is a response map of the filter over the input, indicating the response of that filter pattern at different locations in the input (see figure 5.3). That is what the term feature map means: every dimension in the depth axis is a feature (or filter), and the 2D tensor output[:, :, n] is the 2D spatial map of the response of this filter over the input.”
Convnets

Note that the output width and height may differ from the input width and height. They may differ for two reasons:

- **Border effects**, which can be countered by padding the input feature map. In layer_conv_2d the padding argument, which takes two values: “valid”, “same”
- The use of **strides**. Using stride 2 means the width and height of the feature map are downsampled by a factor of 2.
Max Pooling

The role of max pooling: to **aggressively downsample feature maps**, much like strided convolutions.

“The reason to use downsampling is to reduce the number of feature-map coefficients to process, as well as to induce spatial-filter hierarchies by making successive convolution layers look at increasingly large windows (in terms of the fraction of the original input they cover).”
Training convnets

“Having to train an image-classification model using very little data is a common situation, which you’ll likely encounter in practice if you ever do computer vision in a professional context.”

Classifying images as **dogs or cats**
Training covnets

Strategy to tackle this problem:

- baseline
- data augmentation
- feature extraction with a pretrained network
- fine-tuning a pretrained network
  - Google Seed Bank Try Keras fashion NIST on CoLab.
  - Google TensorFlow Hub
  - Google TensorOrFlow Models
  - tensorspace
Dogs vs Cats

Download the data from kaggle dogs-vs-cats.

Note that you will need to modify the original download directory to the location where you have download the data and unzipped it.

> original_dataset_dir <- "~/Downloads/kaggle_original_data"

Remember in Windows the path needs forward slashes or double back slashes.
“The depth of the feature maps progressively increases in the network (from 32 to 128), whereas the size of the feature maps decreases (from $148 \times 148$ to $7 \times 7$). This is a pattern you’ll see in almost all convnets.”
Convert .jpg into floating-point tensors. Keras has the following function that does this.

```r
> image_data_generator()
> flow_images_from_directory()
> model %>% fit_generator()
```
Overfitting and Data Augmentation

Because we are working with a relatively small dataset, overfitting may be a problem.

We have discussed Dropout and Regularization.

In computer vision problems Data Augmentation is commonly used to fight overfitting.
“Data augmentation takes the approach of generating more training data from existing training samples, by augmenting the samples via a number of random transformations that yield believable-looking images. The goal is that at training time, your model will never see the exact same picture twice. This helps expose the model to more aspects of the data and generalize better.”

“In Keras, this can be done by configuring a number of random transformations to be performed on the images.”

> image_data_generator()