index

June 23, 2022

1 Using Pretrained Networks

1.1 Introduction

In this lesson, you'll start to investigate how to use pretrained networks. Recall that when training neural networks, the model is tuning a huge number of weights: several to dozens at each individual layer. Often the largest limiting factor with these models is the quality and size of the training data you have at hand. As such, adapting a pretrained model from a similar problem context that was trained on a larger dataset can lead to stronger overall models when you have limited training data. For example, in image recognition, the VGG-19 network is commonly used to improve the model performance of CNNs with limited training data. VGG-19 was trained on the ImageNet dataset which contains approximately 1.2 million images. Since the initial bottom layers of a CNN pick up small details with later layers picking up larger and larger features, the initial layers of a well trained network are applicable to other problem domains. Similar pretrained networks exist for other domains such as natural language processing as well. With that, let's take a further look at how transfer learning works in detail.

1.2 Objectives

You will be able to:

- Describe the benefits of using pretrained networks
- Explain how pre-trained neural networks are used for feature extraction
- Explain what "freezing" and "unfreezing" a layer means in a neural network

1.3 Why are pretrained convolutional bases useful?

A commonly used approach when performing deep learning on fairly small image datasets is to use pretrained networks. A pretrained network is a network which was previously ran on a large, general dataset, and saved. The advantage is that the hierarchical features learned by this network can act as a generic model, and can be used for a wide variety of computer vision tasks, even if your new problem involves completely different classes of images.

Recall from earlier that more general features such as edges are detected in earlier layers. Due to this, these convolution layers are highly generic and reusable. Layers that are further down the model extract more abstract concepts, so for new datasets with very different objects to classify, you might want to use only the first layers of the model, and not the entire convolutional base.

1.4 Exampled of pretrained networks

Keras has several pretrained models available. Here is a list of pretrained image classification models. All these models are available in `keras.applications` and were pretrained on the ImageNet dataset, a dataset with 1.4 million labeled images and 1,000 different classes.

- DenseNet
- InceptionResNetV2
- InceptionV3
- MobileNet
- NASNet
- ResNet50
- VGG16
- VGG19
- Xception

You can find an overview here too: https://keras.io/applications/

For each of these pretrained models, you can look at their structure. You can simply import the desired pretrained model, and use it as a function with two arguments: `weights` and `include_top`. Use `'imagenet'` in weights in order to use the weights that were obtained when training on the ImageNet dataset. You can choose to include the top of the model (whether or not to include the fully-connected layer at the top of the network), through the argument `include_top`. Here, we'll have a look at the structure of the MobileNet neural network.

[1]:
```
from keras.applications import MobileNet
conv_base = MobileNet(weights='imagenet',
                      include_top=True)
```

```
Using TensorFlow backend.
```

```
WARNING:tensorflow:From //anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From //anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From //anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From //anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:174: The name
tf.get_default_session is deprecated. Please use
tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From //anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:181: The name tf.ConfigProto is
```

```
deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From //anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:1834: The name
tf.nn.fused_batch_norm is deprecated. Please use
tf.compat.v1.nn.fused_batch_norm instead.

WARNING:tensorflow:From //anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:3445: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
Downloading data from https://github.com/fchollet/deep-learning-
models/releases/download/v0.6/mobilenet_1_0_224_tf.h5
17227776/17225924 [==============================] - 1s 0us/step
```

Note that we are dealing with pretty deep and complex networks here!

[2]:
```
conv_base.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 225, 225, 3)       0
_________________________________________________________________
conv1 (Conv2D)               (None, 112, 112, 32)      864
_________________________________________________________________
conv1_bn (BatchNormalization (None, 112, 112, 32)      128
_________________________________________________________________
conv1_relu (ReLU)            (None, 112, 112, 32)      0
_________________________________________________________________
conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 32)      288
_________________________________________________________________
conv_dw_1_bn (BatchNormaliza (None, 112, 112, 32)      128
_________________________________________________________________
conv_dw_1_relu (ReLU)        (None, 112, 112, 32)      0
_________________________________________________________________
conv_pw_1 (Conv2D)           (None, 112, 112, 64)      2048
_________________________________________________________________
conv_pw_1_bn (BatchNormaliza (None, 112, 112, 64)      256
_________________________________________________________________
conv_pw_1_relu (ReLU)        (None, 112, 112, 64)      0
_________________________________________________________________
conv_pad_2 (ZeroPadding2D)   (None, 113, 113, 64)      0
```

```
_________________________________________________________________
conv_dw_2 (DepthwiseConv2D)  (None, 56, 56, 64)        576
_________________________________________________________________
conv_dw_2_bn (BatchNormaliza (None, 56, 56, 64)        256
_________________________________________________________________
conv_dw_2_relu (ReLU)        (None, 56, 56, 64)        0
_________________________________________________________________
conv_pw_2 (Conv2D)           (None, 56, 56, 128)       8192
_________________________________________________________________
conv_pw_2_bn (BatchNormaliza (None, 56, 56, 128)       512
_________________________________________________________________
conv_pw_2_relu (ReLU)        (None, 56, 56, 128)       0
_________________________________________________________________
conv_dw_3 (DepthwiseConv2D)  (None, 56, 56, 128)       1152
_________________________________________________________________
conv_dw_3_bn (BatchNormaliza (None, 56, 56, 128)       512
_________________________________________________________________
conv_dw_3_relu (ReLU)        (None, 56, 56, 128)       0
_________________________________________________________________
conv_pw_3 (Conv2D)           (None, 56, 56, 128)       16384
_________________________________________________________________
conv_pw_3_bn (BatchNormaliza (None, 56, 56, 128)       512
_________________________________________________________________
conv_pw_3_relu (ReLU)        (None, 56, 56, 128)       0
_________________________________________________________________
conv_pad_4 (ZeroPadding2D)   (None, 57, 57, 128)       0
_________________________________________________________________
conv_dw_4 (DepthwiseConv2D)  (None, 28, 28, 128)       1152
_________________________________________________________________
conv_dw_4_bn (BatchNormaliza (None, 28, 28, 128)       512
_________________________________________________________________
conv_dw_4_relu (ReLU)        (None, 28, 28, 128)       0
_________________________________________________________________
conv_pw_4 (Conv2D)           (None, 28, 28, 256)       32768
_________________________________________________________________
conv_pw_4_bn (BatchNormaliza (None, 28, 28, 256)       1024
_________________________________________________________________
conv_pw_4_relu (ReLU)        (None, 28, 28, 256)       0
_________________________________________________________________
conv_dw_5 (DepthwiseConv2D)  (None, 28, 28, 256)       2304
_________________________________________________________________
conv_dw_5_bn (BatchNormaliza (None, 28, 28, 256)       1024
_________________________________________________________________
conv_dw_5_relu (ReLU)        (None, 28, 28, 256)       0
_________________________________________________________________
conv_pw_5 (Conv2D)           (None, 28, 28, 256)       65536
_________________________________________________________________
conv_pw_5_bn (BatchNormaliza (None, 28, 28, 256)       1024
```

```
_________________________________________________________________
conv_pw_5_relu (ReLU)        (None, 28, 28, 256)       0
_________________________________________________________________
conv_pad_6 (ZeroPadding2D)   (None, 29, 29, 256)       0
_________________________________________________________________
conv_dw_6 (DepthwiseConv2D)  (None, 14, 14, 256)       2304
_________________________________________________________________
conv_dw_6_bn (BatchNormaliza (None, 14, 14, 256)       1024
_________________________________________________________________
conv_dw_6_relu (ReLU)        (None, 14, 14, 256)       0
_________________________________________________________________
conv_pw_6 (Conv2D)           (None, 14, 14, 512)       131072
_________________________________________________________________
conv_pw_6_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_6_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_dw_7 (DepthwiseConv2D)  (None, 14, 14, 512)       4608
_________________________________________________________________
conv_dw_7_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_7_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_7 (Conv2D)           (None, 14, 14, 512)       262144
_________________________________________________________________
conv_pw_7_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_7_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_dw_8 (DepthwiseConv2D)  (None, 14, 14, 512)       4608
_________________________________________________________________
conv_dw_8_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_8_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_8 (Conv2D)           (None, 14, 14, 512)       262144
_________________________________________________________________
conv_pw_8_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_8_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_dw_9 (DepthwiseConv2D)  (None, 14, 14, 512)       4608
_________________________________________________________________
conv_dw_9_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_9_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_9 (Conv2D)           (None, 14, 14, 512)       262144
```

```
_________________________________________________________________
conv_pw_9_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_9_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_dw_10 (DepthwiseConv2D) (None, 14, 14, 512)       4608
_________________________________________________________________
conv_dw_10_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_10_relu (ReLU)       (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_10 (Conv2D)          (None, 14, 14, 512)       262144
_________________________________________________________________
conv_pw_10_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_10_relu (ReLU)       (None, 14, 14, 512)       0
_________________________________________________________________
conv_dw_11 (DepthwiseConv2D) (None, 14, 14, 512)       4608
_________________________________________________________________
conv_dw_11_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_11_relu (ReLU)       (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_11 (Conv2D)          (None, 14, 14, 512)       262144
_________________________________________________________________
conv_pw_11_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_11_relu (ReLU)       (None, 14, 14, 512)       0
_________________________________________________________________
conv_pad_12 (ZeroPadding2D)  (None, 15, 15, 512)       0
_________________________________________________________________
conv_dw_12 (DepthwiseConv2D) (None, 7, 7, 512)         4608
_________________________________________________________________
conv_dw_12_bn (BatchNormaliz (None, 7, 7, 512)         2048
_________________________________________________________________
conv_dw_12_relu (ReLU)       (None, 7, 7, 512)         0
_________________________________________________________________
conv_pw_12 (Conv2D)          (None, 7, 7, 1024)        524288
_________________________________________________________________
conv_pw_12_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_pw_12_relu (ReLU)       (None, 7, 7, 1024)        0
_________________________________________________________________
conv_dw_13 (DepthwiseConv2D) (None, 7, 7, 1024)        9216
_________________________________________________________________
conv_dw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_dw_13_relu (ReLU)       (None, 7, 7, 1024)        0
```

```
_________________________________________________________________
conv_pw_13 (Conv2D)          (None, 7, 7, 1024)        1048576
_________________________________________________________________
conv_pw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_pw_13_relu (ReLU)       (None, 7, 7, 1024)        0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 1024)              0
_________________________________________________________________
reshape_1 (Reshape)          (None, 1, 1, 1024)        0
_________________________________________________________________
dropout (Dropout)            (None, 1, 1, 1024)        0
_________________________________________________________________
conv_preds (Conv2D)          (None, 1, 1, 1000)        1025000
_________________________________________________________________
reshape_2 (Reshape)          (None, 1000)              0
_________________________________________________________________
act_softmax (Activation)     (None, 1000)              0
=================================================================
Total params: 4,253,864
Trainable params: 4,231,976
Non-trainable params: 21,888
_________________________________________________________________
```

You'll learn about two ways to use pre-trained networks: - **Feature extraction**: Here, you use the representations learned by a previous network to extract interesting features from new samples. These features are then run through a new classifier, which is trained from scratch.

- **Fine-tuning**: When fine-tuning, you'll "unfreeze" a few top layers from the model and train them again together with the densely connected classifier. Note that you are changing the parts of the convolutional layers here that were used to detect the more abstract features. By doing this, you can make your model more relevant for the classification problem at hand.

1.5 Feature Extraction

Feature extraction with convolutional neural networks means that you take the convolutional base of a pretrained network, run new data through it, and train a new classifier on top of the output (a new densely connected classifier). Why use convolutional base but *new* dense classifier? Generally, patterns learned by the convolutional layers are more generalizable.

Note that, if your dataset differs a lot from the dataset used when pretraining, it might even be worth only using part of the convolutional base (see "fine tuning")

Also, with feature extraction, there are two ways running the model: - You can run the convolutional base over your dataset, save its output, then use this data as input to a standalone, densely connected network. This solution is pretty fast to run, and you need to run the convolutional base first for every input image. The problem here is, however, that you can't use data augmentation as we've seen before. - You can extend the convolutional base by adding dense layers on top, and running everything altogether on the input data. This way, you can use data augmentation, but as every input image goes through the convolutional base every time, this technique is much more

time-consuming. It's almost impossible to do this without a GPU

1.6 Fine tuning

Fine tuning is similar to feature extraction in that you reuse the convolution base and retrain the dense, fully connected classifier layers to output a new prediction. In addition, fine tuning also works by retraining the frozen weights for the convolutional base. This allows these weights to be tweaked for the current scenario, hence the name. To do this, you'll freeze part of the model while tuning specific layers.

1.7 Additional Resources

- http://cs231n.stanford.edu/syllabus.html
- https://www.dlology.com/blog/gentle-guide-on-how-yolo-object-localization-works-with-keras/
- https://www.dlology.com/blog/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-2/

1.8 Summary

In this lesson we introduced the concept of using pretrained neural networks as a starting point for future models. The idea is that well trained CNNs have generalized visual patterns that are transferable to new applications. For example, a previously trained model might have hidden convolutional layers that effectively identify edges, or even more complex patterns such as eyes which may have value in new domains. In the upcoming lab, we'll further explore how to implement these concepts in Keras.