Image Classification of Outdoor Scenes with Convolutional Neural Networks

Adam Young 7/27/2021

Problem Statement:

Image classification problems are common in many different areas of business and research. Self-driving cars use image classification to detect pedestrians and other important signals on the road, cancer treatment centers use image classification to efficiently detect tumors in body scans, and tech giants like Facebook use image classification to identify which friend is standing next to you in the recent photo that you posted. The applications are endless, but for this project the challenge is to classify pictures of outdoor landscapes into 6 different categories: buildings, mountains, glaciers, streets, forests, and seas.

Data Description and Import

Data for this project comes from the Intel Image Classification dataset and includes 14,034 training images and 3,000 test images. The training and test set were both balanced, containing roughly the same number of images of each category. Each image is located within a folder labeled as one of the 6 different classes. To import the data and get it ready for modeling, the following steps needed to be executed:

1. Load each image independently
2. Resize the image to 150x150 pixels
3. Convert the image to an array of pixel values
4. Scale the pixel values between 0 and 1
5. Append the image array to the dataset
6. Append the image's class to the list of labels

The resulting dataset included a training image array with shape (14034, 150, 150, 3) and a list of the corresponding 14,034 training labels.

Initial Modeling

Initial testing was done on a basic convolutional neural network containing 3 ReLu-activated convolutional layers, each followed by max pooling layers, and finally a dense layer on top. The model summary can be seen below.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 36992)             0
_________________________________________________________________
dense (Dense)                (None, 64)                2367552
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 390
=================================================================
Total params: 2,461,190
Trainable params: 2,461,190
Non-trainable params: 0
_________________________________________________________________
```

Training and evaluating this basic model resulted in an accuracy of 79.80% on the validation set. To improve this accuracy, many variations of the model were tested:

Model	Description	Val. Accuracy
1	20 epochs, max pooling, no image augmentation	79.80%
2	20 epochs, max pooling, image augmentation (width shift=0.2, horizontal flip=True, zoom range=0.2, shear range=0.2)	86.30%
3	20 epochs, max pooling, image augmentation (width shift=0.2, horizontal flip=True, zoom range=0.2, shear range=0.2, height shift = 0.2, rotation range=0.2)	84.90%
4	20 epochs, max pooling, image augmentation (width shift=0.4, horizontal flip=True, zoom range=0.4, shear range=0.4)	87.40%
5	20 epochs, max pooling, image augmentation (width shift=0.4, horizontal flip=True, zoom range=0.4, shear range=0.4, height shift = 0.4, rotation range=0.4)	84.13%
6	100 epochs, max pooling, image augmentation (width shift=0.2, horizontal flip=True, zoom range=0.2, shear range=0.2)	86.50%
7	20 epochs, average pooling, image augmentation (width shift=0.2, horizontal flip=True, zoom range=0.2, shear range=0.2)	85.20%
8	100 epochs, max pooling, image augmentation (width shift=0.4, horizontal flip=True, zoom range=0.4, shear range=0.4)	87.47%
9	100 epochs, max pooling, image augmentation (width shift=0.4, horizontal flip=True, zoom range=0.4, shear range=0.4, channel shift=0.4)	87.03%

The best performing model had an accuracy of 87.47% on the test set. The model architecture was the same as the initial model, but it was trained for 100 epochs, and utilized image augmentation techniques including width shift, horizontal flip, zoom, and shear. Image augmentation is a technique that enables you to generate additional data samples which are simply augmented variations of the original data. This technique is useful because it provides the model with additional training samples, and convolutional neural networks are typically very sensitive to the size of the training data, meaning that they perform much better when given more data to train on. Note that not all image augmentation techniques should be used in every circumstance. For example, vertical shift and vertical flip were not used in this case because they did not make sense in the context of the dataset. A building or mountain will never be upside down.

Transfer Learning

Transfer learning is a process by which a model that has already been trained on a different dataset is then used to make classifications for a new problem. The basic process is as follows:

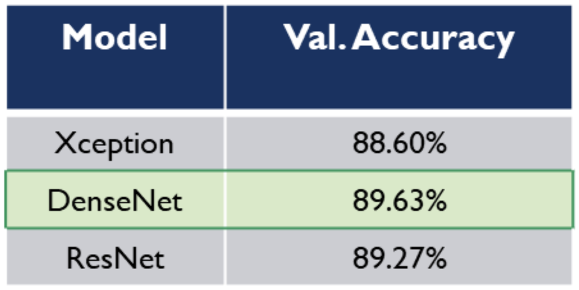
1. Select a model to use for transfer learning
2. Import the model
3. Freeze the model, in order to keep the features and weights that it has already learned
4. Add one or multiple fully connected layers on top of the existing model
5. Train only those fully connected layers on the applicable dataset
6. (Optional) Unfreeze some of the pre-trained layers and train them

The most common models used for transfer learning are those that have been trained on the ImageNet database. This dataset is chosen for a few main reasons:

- Size: the ImageNet database has over 14 million labeled images
- Diversity: the ImageNet database has over 21 thousand different image classes

- Popularity: The ImageNet dataset has been used by many of the largest tech companies and research institutions in the world to train many different types of neural networks

In the interest of time, only 3 of the top performing models on the ImageNet database were evaluated on the outdoor image classification problem. These were Xception, DenseNet and ResNet50.



Model	Val. Accuracy
Xception	88.60%
DenseNet	89.63%
ResNet	89.27%

Utilizing transfer learning increased the accuracy up to 89.63% for a 2.16% increase over the basic model.

Takeaways and Future Improvements

Using a basic CNN architecture, the maximum accuracy achieved was 87.47%. It's possible that with some additional experimentation this number could be boosted up by a percent or two, but a much easier way to increase performance is to implement transfer learning. Instead of fiddling around with the model architecture, a pre-trained model architecture can be loaded and used on (almost) any classification task. The main reason that transfer learning would result in poor performance is if the images that need to be classified are too dissimilar from the images that the transfer model was trained on. The ImageNet dataset is diverse enough that models trained on it will perform well on most image classification tasks, but it's possible that a very unique image classification problem would not work well with an ImageNet based model. For this problem the DenseNet transfer learning model was the highest performing, with an accuracy of 89.63%.

With additional time and computational resources, it would be interesting to attempt to fine-tune the transfer learning models. This is done by un-freezing the top layers of the model and re-training them on the new image dataset. The goal of this is to better adjust the transfer model to the new data, apart from just fitting dense layers on top of it.