Machine Learning Engineer Nanodegree - Capstone Proposal

Intel Scene Classification with CNN

Shibin Menachery
August 16th, 2019

Domain Background

The amount of visual data has increased exponentially over the past few years. Running ML models to extract certain features like pedestrians, cars etc. without knowing if the image is an eligible candidate can result in sending large amounts of images without the required background and hence wasting a lot of computing resources. For example, we have a ML model that detects restaurants and shops in images, however we keep sending crowdsourced images of nature with no buildings in it. Hence to overcome this, in this proposal I intend to create a scene classification ML model that given a predefined set of scene categories, will classify any unseen image into one of the scene categories.

In this study, a deep learning model based on CNNs is proposed for the Intel Image Classification Challenge from Kaggle. There are other approaches other than the CNN based ones like the bag of words (Rasiwasia & Vasconcelos, Scene Classification with Low-dimensional Semantic Spaces and Weak Supervision) model.

My personal motivation to solve this problem is due to one of the scenarios I am tackling at work where I am encountering a large influx of images which needs to be processed to extract customer specific attributes from them. I found that a lot of my compute time is wasted working on images which are not related to the attributes what the customer specifies. If I can pre assign images with scene labels then this will help me reduce the number images to be sent forward for further processing thus saving on costs and time.

Problem Statement

Due to the increasing amount of visual data of image classification solutions of images generated by various sources. It is not possible for a human annotator to sift through this huge influx of image and categorize them appropriately to be used further. We investigate how to train a CNN that can classify scenes exploring developing one from scratch as well as using transfer learning.

Datasets and Inputs

The dataset is provided by the Intel scene classification challenge. The dataset contains:
Image Size: 150x150
Training Set: 17034 images (To be split into training, validation and test) **Pred Set**: 7032 images
Categories:

Scene	**Label**	**Image Distribution**
buildings	0	2628
forest	1	2745
glacier	2	2957
mountain	3	3037
sea	4	2784
street	5	2883

The images for each label in training is not evenly distributed and class distribution varies from 2628 to 3037. This demonstrates that the data is imbalanced and the data need to be balanced in order to get the best results.

Also while sampling some images I saw it had a combination of scenes but the label was there for only one . It would be interesting to see how this plays out during training.

There are 3 files provided: train.zip which contains all images, train.csv and test.csv which has the following structure:

Key	**Value**
Image_name	Name of the image in the dataset
Label	Category of natural scene

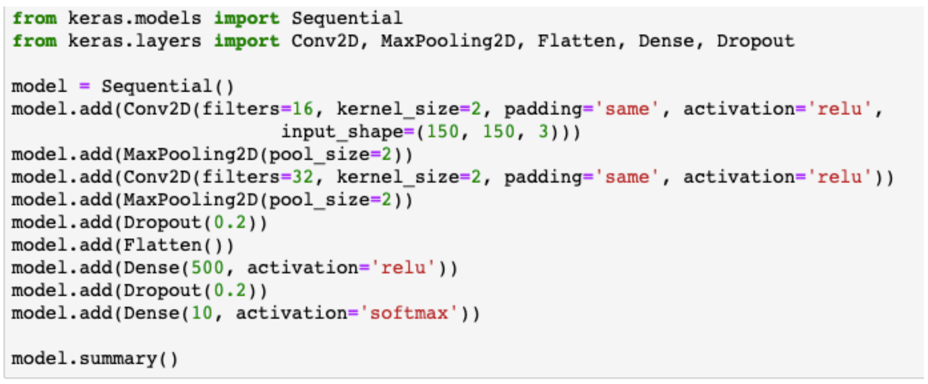
Source: https://www.kaggle.com/nitishabharathi/scene-classification

Solution Statement

The CNN model will be built using TensorFlow with Keras library. The CNN will employ the convolutional layers, pooling layers, and fully connected layers. We will use different strides and padding values and evaluate the output using ROC curves. We will also evaluate Regularization techniques such as dropout, batch normalization and data augmentation too if it improves the accuracy which will be used as the model performance metric. Once the above methods have exhausted with our limited training data and computing power we will explore Transfer learning and evaluate pre-trained models such as Resnet50, Resnet101 etc.

Benchmark Model

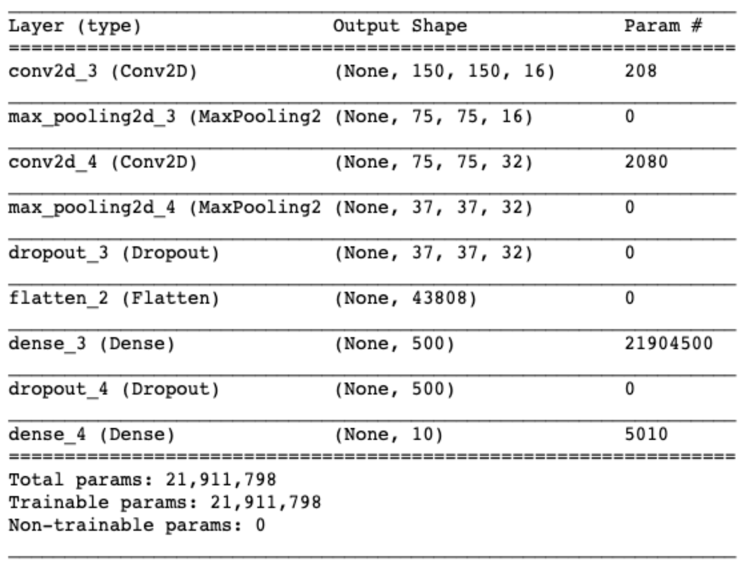
To create a benchmark for the CNN models I will build, I will take my first model (a simple CNN model) trained with the same dataset as the base model. Then I will work on it by tweaking various parameters such as Conv2D, MaxPooling2D, Flatten, Dense, Dropout and then further use various loss functions and optimizers to reach a solution model. The first model and the solution model will be compared based on the accuracy. A simple CNN vanilla model will have a structure as follows:



```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential()
model.add(Conv2D(filters=16, kernel_size=2, padding='same', activation='relu',
                        input_shape=(150, 150, 3)))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()
```



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            (None, 150, 150, 16)      208       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 75, 75, 16)        0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 75, 75, 32)        2080      
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 37, 37, 32)        0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 37, 37, 32)        0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 43808)             0         
_________________________________________________________________
dense_3 (Dense)              (None, 500)               21904500  
_________________________________________________________________
dropout_4 (Dropout)          (None, 500)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 10)                5010      
=================================================================
Total params: 21,911,798
Trainable params: 21,911,798
Non-trainable params: 0
_________________________________________________________________
```

Evaluation Metrics

The overall evaluation metric that will be used for the benchmark model and the solution model is accuracy:

accuracy = Number of correctly predicted class/Total number of predictions

Then we will use a confusion matrix to evaluate our model to see if the accuracy is skewed to a particular class or not as we know that our dataset is imbalanced. If in the confusion matrix we see that classes with less data are indeed having less scores then we will take steps such as undersampling, oversampling or creating PR curves to evaluate our model.

1. Under sampling by removing samples from over-represented classes.
2. Oversampling: SMOTE(Synthetic Minority Over-sampling Technique) to create synthetic samples of the minority class.
3. PR(Precision-Recall) curves, if we are keeping the dataset as it is. High area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate.

Project Design

We will approach the project by following the below workflow:

- Building CNN model from scratch
  - Load the dataset and visualize the images
  - Rescale the images by dividing every pixel by 255
  - Break training dataset into training testing and validation sets
  - Use Conv2D, MaxPooling2D, Flatten, Dense, Dropout to define model
  - Use loss function(*categorical_crossentropy*) and optimizers(*adam, sgd, rmsprop*) to compile the model
  - Train, checkpoint and save the weights
  - Predict, evaluate and record the accuracy
  - Visualize the test data and ground truth
  - Go to first step again and change

- Using Transfer Learning
  - Pick Resnet50 model pretrained on ImageNet
  - Set include_top to False
  - Follow steps similar to one defined for building CNN model from scratch
  - Save the results and then similarly evaluate Resnet101