Machine Learning Engineer Nano degree

Capstone Project

Mustafa Mahmoud abd Elkawy

September 21st, 2018

I. Definition

Project Overview

With the recent huge advances in technology, the importance of visual recognition and image processing has significantly increased. As Deep Learning and Computer Vision help us better understand images and extract useful information out of them. In the field of Self-driving cars, healthcare and pretty much every field that deals with images, visual recognition is proven to be of high importance.

For this project I created a classifier capable of taking an image of Fruit and predict its respective category. This task achieved by a deep neural network. This is part of a more complex project that has the target of obtaining a classifier that can identify a much wider array of objects from images. This fits the current trend of companies working in the augmented reality field.

During its annual I/O conference, Google announced that is working on an application named Google Lens which will tell the user many useful information about the object toward which the phone camera is pointing. First step in creating such application is to correctly identify the objects. Currently the identification of objects is based on a deep neural network.

Such a network would have numerous applications across multiple domains like autonomous navigation, modeling objects, controlling processes or human-robot interactions.

The project was inspired by : this research paper.

Problem Statement

Fruits have certain categories that are hard to differentiate, like the citrus genus that contains oranges and grapefruits. Thus I want to see how well an artificial intelligence can complete the task of classifying them.

The goal of the project is to make classifier capable of taking an image of Fruit and predict its respective category. The tasks involved are the following:

- Step 0: Download the data set.
- Step 1: take a subset of the training data set to be used as validation set.
- Step 2: Import Datasets
- Step 3: preprocessing steps supply images to a pre-trained network in Keras
- Step 4: Extract Bottleneck Features for Train set, valid set, Test Set.
- Step 5: rescale the images by dividing every pixel in every image by 255.

- Step 6: Obtain Bottleneck Features.
- Step 7: create Model Architecture.
- Step 8: Train the Model.
- Step 9: Test the Model.
- Step 10: Test the Model on Sample Images!

Metrics

I evaluated my model using accuracy score test on the test set to check the accuracy of my model.

$$Accuracy = \frac{true\ positives + true\ negatives}{dataset\ size}$$

this metric was used when evaluating the classifier because There is no imbalance in the data.

II. Analysis

Data Exploration

In this project, I used this data set on kaggle , data set on GitHub it contains 55244.images of Fruits. It was the main motive for the project. It was collected by Mihai Oltean.

Here's the content of the dataset. It

- Total number of images: 55244.
- Training set size: 41322 images (one fruit per image).
- Test set size: 13877 images (one fruit per image).

- Multi-fruits set size: 45 images (more than one fruit per image)
- Number of classes: 81 (fruits).
- File format: standard RGB images
- Image size: 100x100 pixels.

Only problem with the dataset:

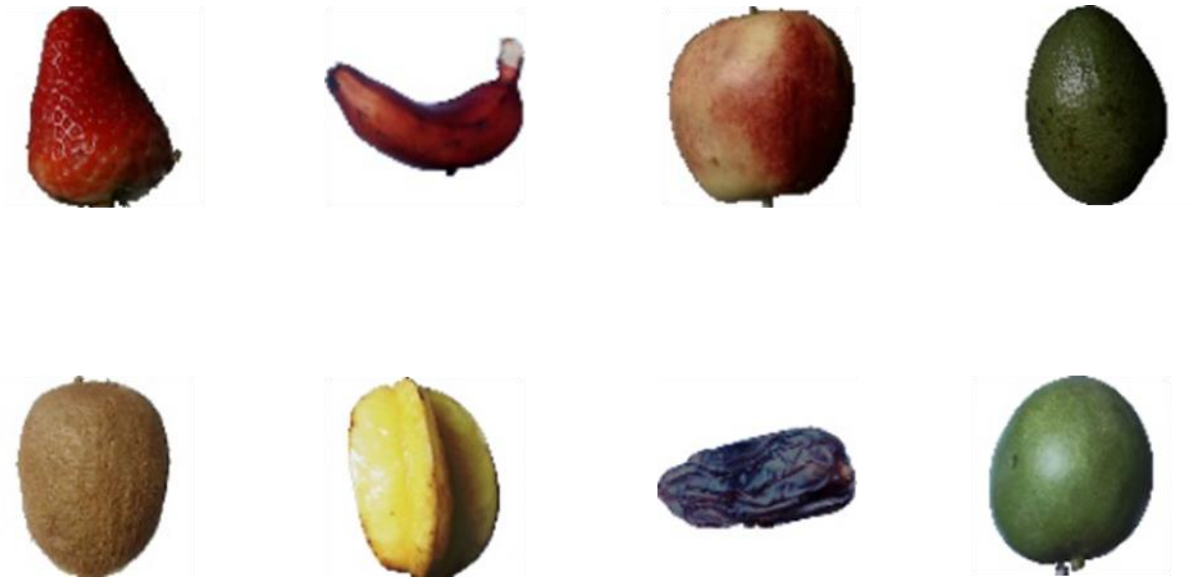
Unfortunately the dataset have only two subsets. A subset for training and a subset for test. So I had to take a subset of training subset to be used as validation set.

My version of the data after take a subset of the training data to be used as validation data. used_data.

Here's the content of my version of the dataset. It

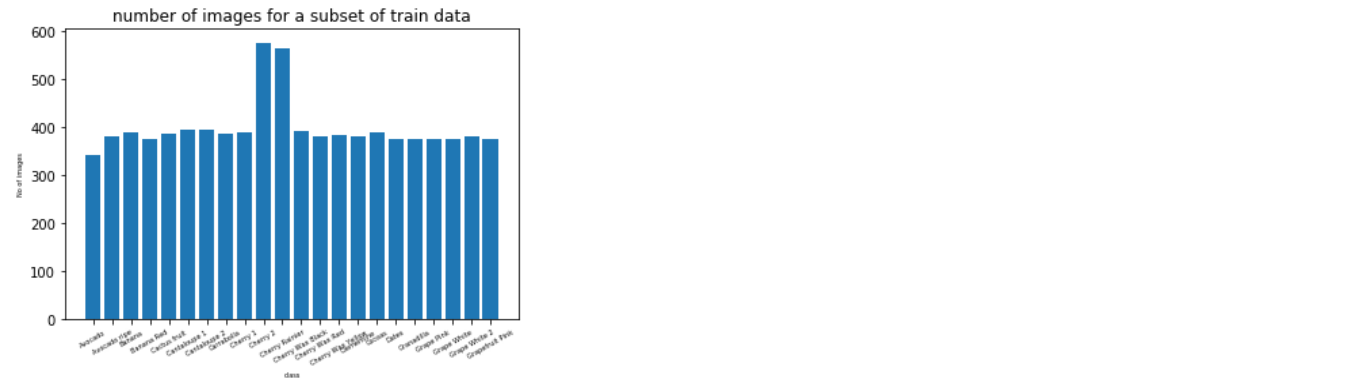
- Total number of images: 55199.
- Training set size: 31983 images (one fruit per image).
- Validation set size: 9339 images (one fruit per image).
- Test set size: 13877 images (one fruit per image).
- Number of classes: 81 (fruits).
- File format: standard RGB images
- Image size: 100x100 pixels.

Preview for the content of the dataset:



Exploratory Visualization

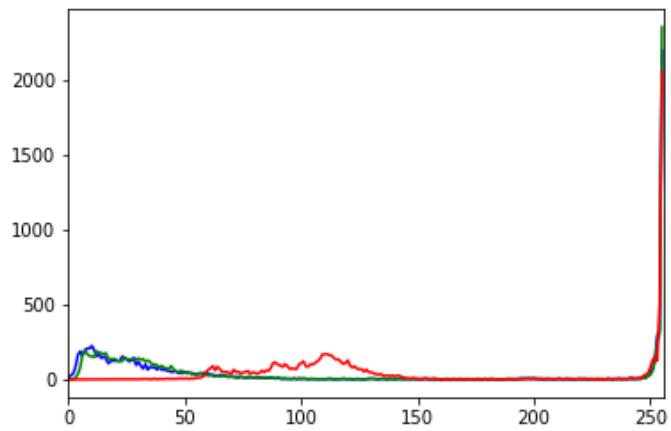
The figure below shows the number of images in the subset of training set as you see each class has almost the same number of images which is good because very big difference in number of image per class the model will suffer from biased to the biggest class.



The below table showing for instance (subset of the data) that the target variable distributed as:

Label	Number of training images	Number of valid images	Number of test images
Avocado	341	86	143
Avocado ripe	381	110	166
Banana	390	100	166
Banana Red	376	114	166
Cactus fruit	385	105	166
Cantaloupe 1	393	99	164
Cantaloupe 2	394	98	164
Carrabolla	385	105	166
Cherry 1	389	103	164
Cherry 2	576	162	246
Cherry Rainier	564	174	246
Cherry Wax Black	392	100	164
Cherry Wax Red	380	112	164
Cherry Wax Yellow	382	110	164
Clementine	380	110	166
Cocoas	390	100	166
Dates	375	115	166
Granadilla	375	115	166
Grape Pink	374	118	164
Grape White	374	116	166
Grape White 2	380	110	166
Grapefruit Pink	374	116	166

In the figure below, I tried to get a glimpse of how the pixels intensity looks like by creating a histogram to show that.



Algorithms and Techniques

The classifier is a Convolutional Neural Network which is the state-of-the-art algorithm for most image processing tasks, including classification. It needs a large amount of training data compared to other approaches; fortunately, dataset are big enough. The algorithm outputs an assigned probability for each class. This is very good because using a classification threshold, the number of false positives can be reduced.

- The tradeoff is this increases the number of false positives.

The following parameters can be tuned to optimize the classifier:

- The classification threshold (as mentioned above)
- Solver (what algorithm to use for Transfer learning).
- Training length (number of epochs).
- Batch size (how many images to look at once during a single training step).
- Neural Network Architecture
- Number of Layers
- Layer Types (Convolutional, fully connected or pooling)

I used Transfer Learning which focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

For these types of problems, it is common to use a deep learning model pre-trained for a large and challenging image classification task such as the ImageNet 1000-class photograph classification competition. In my case Xception model was used.

Input:

When using Tensor Flow as backend, Keras CNNs require a 4D array (which we'll also refer to as a 4D tensor) as input, with shape

(nb_samples, rows, columns, channels)

Where `nb_samples` corresponds to the total number of images (or samples), and `rows`, `columns`, and `channels` correspond to the number of rows, columns, and channels for each image, respectively.

Benchmark

I have benchmarked my model against the results presented in this research paper.

The structure of the neural network used in the paper:

Layer type	Dimensions	Output
Convolutional	5 x 5 x 4	16
Max pooling	2 x 2 — Stride: 2	-
Convolutional	5 x 5 x 16	32
Max pooling	2 x 2 — Stride: 2	-
Convolutional	5 x 5 x 32	64
Max pooling	2 x 2 — Stride: 2	-
Convolutional	5 x 5 x 64	128
Max pooling	2 x 2 — Stride: 2	-
Fully connected	5 x 5 x 128	1024
Fully connected	1024	256
Soft max	256	60

The calculated accuracy of the neural network used in the paper was 96.19%.

III. Methodology

Data Preprocessing

The preprocessing done in the “preprocessing steps supply images to a pre-trained network in Keras” notebook consists of the following steps:

The path_to_tensor function takes a string-valued file path to a color image as input and returns a 4D tensor suitable for supplying to a Keras CNN. The function first loads the image and resizes it to a square image that is 100×100 pixels. Next, the image is converted to an array, which is then resized to a 4D tensor.

In this case, since I am working with color images, each image has three channels. Likewise, since I am processing a single image (or sample), the returned tensor will always have shape

(1, 100, 100, 3)

The paths_to_tensor function takes a numpy array of string-valued image paths as input and returns a 4D tensor with shape

(nb_samples, 100, 100, 3)

Here, nb_samples is the number of samples, or number of images, in the supplied array of image paths. It is best to think of nb_samples as the number of 3D tensors (where each 3D tensor corresponds to a different image) in my dataset!

Additional preprocessing done in the “rescale the images” notebook by dividing every pixel in every image by 255.

Implementation

The implementation process can be split into two main stages:

1. Extract Bottleneck Features for Train set, valid set, and Test Set stage.
2. The classifier training stage

During the first stage Bottleneck Features for Train set, valid set, and Test Set Extracted from the preprocessed data. This was done in a Jupiter notebook (titled “fruit_app”), and can be further divided into the following steps:

1. Load images into memory, preprocessing them as described in the previous section.
2. Load model that used for Transfer learning
3. Check if the Bottleneck Features file is exist or not.
4. If the file is not exist calculate the features of the data by predict the preprocessed data.
5. Save the features to .npz file.
6. Repeat steps from 3 to 5 for validation, test sets.

The second stage can be further divided into the following steps:

1. Obtain Bottleneck Features

   Load the features from .npz files to the memory.

2. create Model Architecture:

   I used a standard Architecture that contain of two layers:

a. GlobalAveragePooling2D

   Take the training features shape as input shape.

b. Dense

   The output layer with 81 node.

3. Define the loss function, accuracy.

4. Train the network, logging the validation/training loss and the validation accuracy.
5. If the accuracy is not high enough, return to stage 1 and choose another model for transfer learning.

6. Save and freeze the trained network.

The coding process went smoothly, except the part with Extract Bottleneck Features technology that I was not familiar with before the project. This got me frustrated a lot and I had to do a lot of research.

Refinement

In fact, there was a small improvement process as mentioned in the implementation section in the first phase. I had to try more than one model to use in the transfer learning process and make the most of the technique to improve accuracy. My last choice to use Xception instead of models like VGG-19, ResNet-50 and Inception was no surprise to me because it came recently compared to others.

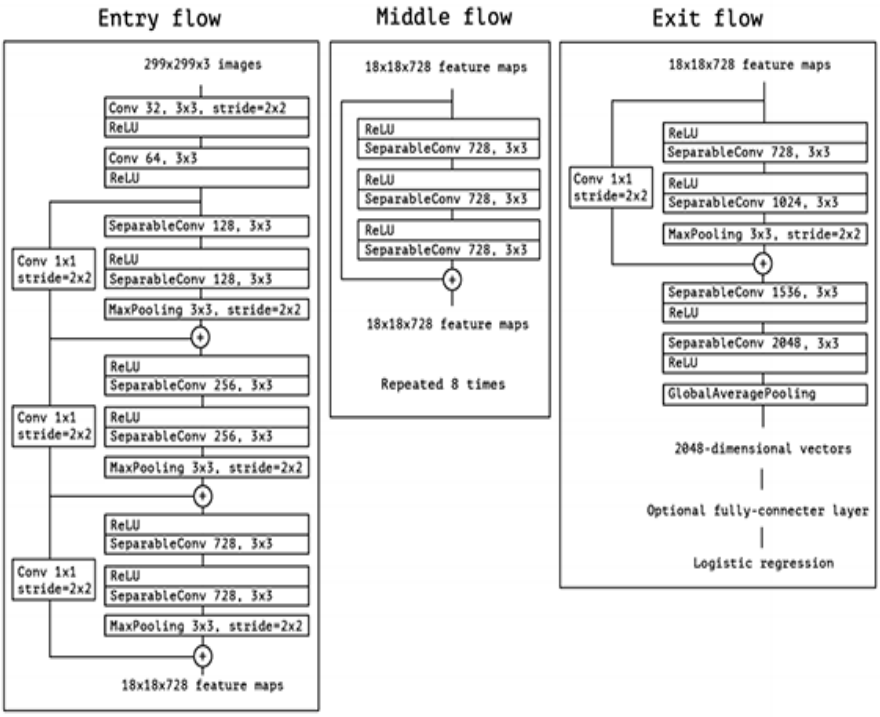
IV. Results

Model Evaluation and Validation

During development, a validation set was used to evaluate the model.

The final architecture and hyper parameters were chosen because they performed well.

The structure of the model used for transfer learning:



My final model architecture:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_1 ( (None, 2048)              0

_________________________________________________________________
dense_1 (Dense)              (None, 81)                165969
=================================================================
Total params: 165,969
Trainable params: 165,969
Non-trainable params: 0

_________________________________________________________________
```

My Final test accuracy is 95.5178%

To verify the robustness of the final model, a test was conducted using images of deferent fruits (see Free-Form Visualization section for instance).

The following observations are based on the results of the test:

- The classifier can reliably detect one fruit per image.
- Images have to be clear (no background).
- The classifier can't detect more than one fruit per image.

Justification

My Final test accuracy is 95.5178% which is very close of my benchmark model accuracy.

It can be seen that the application is useful for detect one fruit per image. But also that it can get confused by more than one fruit per image, and it can miss fruits that is too low-contrast compared to its background.

In summary, the application is useful in a limited domain, but to solve the bigger problem. Additional data and steps will have to be used (see Improvement section).

V. Conclusion

Free-Form Visualization

Fig. 1 Examples of fruits classified by the classifier.

it look like a Strawberry Wedge

it look like a Orange

it look like a Banana Red

it look like a Pitahaya Red

it look like a Grape Pink

it look like a Mandarine

it look like a Mulberry

it look like a Physalis

it look like a Apple Red Delicious

From Figure 1, three failure cases can be clearly identified:

1. More than one fruit per image.
2. Fruits that is too low-contrast compared to its background.
3. Very similar fruits such as bananas and red bananas.

Reflection

The process used for this project can be summarized using the following steps:

1. An initial problem and relevant, public datasets were found.
2. The data was downloaded and modified by take a subset of training set to be used as validation test .and have been uploaded to my github to be accessible.
3. A benchmark was found.
4. The data was preprocessed.
5. Extract Bottleneck Features for Train set, valid set, and Test Set.
6. The classifier was trained using transfer learning technique on the data.
7. The classifier was tested.

I found steps 2 and 5 the most difficult, as I had to take a subset of the data manually to be sure of data balancing. And Extract Bottleneck Features technology that I was not familiar with before the project.

As for the most interesting aspects of the project, I'm very glad that I found the Fruits-360 data set, as I'm sure they'll be useful for later projects/experiments. I'm also happy about getting to use transfer learning technique, as I believe it will be very useful in the future.

Improvement

As mentioned earlier there was some failure cases but to solve the bigger problem. I think that dataset have to be expanded with more:

1. Images for Very similar fruits such as bananas and red bananas.
2. Images for more than one fruit per image.

Another note that have to be considered as a way of improvement that the image have to be clean(extract fruit from the background) before test the classifier on it because the dataset were collected and preprocessed by algorithm to extract the fruit from the background.
