Machine Learning Engineer Nanodegree

Capstone Project

Augustas Volbekas

2017-05-24

I. Definition

Project Overview

Self service checkout systems are commonly found in retail stores. These machines are a great way to minimize waiting time. Using them is easy – scan the barcode. However not all products have barcodes for example fresh fruit and vegetables, in self service systems these products have to be found by category and selected from a range of similar options. This could be a hassle for a customer not used to the system. This project aims to provide a solution to the problem using machine learning. The basic design would be, to have a camera over the top to take pictures and classify then provide suggestions for the based on classification. This is a “proof of concept” project.

Problem Statement

The problem is in the field of computer vision and supervised machine learning. It is a classification of pictures with items placed in a constant background. The model should be able to classify a range of common groceries given a picture.

Inputs would be pictures and labels would the items in the picture.

The possible use case of the model would be to help customers use a self service checkout systems in retail stores by providing suggestions of items placed on the scales.

The algorithm used is a convolutional neural network.

As this is a “proof of concept” project the data used a pictures of 6 grocery items in a white background. A total of 3000 pictures.

Metrics

To measure the model a simple accuracy metric is used. Accuracy – number of accurate classifications, divided by total number of classifications.

This simple metric would clearly show how well the model is classifying.

II. Analysis

Data Exploration

The data set - pictures used are made in a home setting. All pictures have a shape of (1024,1024,3). There are 3000 pictures of common grocery items. The pictures are divided to 6 categories, 500 pictures each:

Pumpkin

Lemon

Cucumbers

Potatoes

Pumpkin seeds

Onions

Pictures contain items in a bag, one item or multiple items. Also hands are present in some pictures and some pictures have items in motion.

The whole data set could be accessed with a Dropbox link:

https://www.dropbox.com/s/b2kdr4d7pw6095d/cashier.tar.gz?dl=0

Exploratory Visualization

Some samples of pictures used.






Algorithms and Techniques

A convolutional neural network was used, because the model can work with item being in any position in the picture also convolutional layers require less parameters to train than it would be needed for a fully connected neural network, given the large size of the input.

The input to the net is a picture as a multidimensional array of pixel RGB values.

The basic design of the model is a few iterations of a convolutional layer followed by max pooling layer, followed by fully connected layers.

The model parameters were changed many times in order to find optimal. Attempt was made to make a model model to have a little parameters to train, but also retain a good accuracy score.

The architecture of the model:

2D_convolution layer: size – (3,3), features – 32, strides (2,2), activation - “relu”

Max_pool layer: size - (3,3), strides – (3,3)

2D_convolution layer: size – (3,3), features – 64, strides (2,2), activation - “relu”

Max_pool layer: size - (3,3), strides – (3,3)

Dropout: 0.25

Fully connected layer: nodes – 64, activation - “relu”

Dropout: 0.25

Fully connected layer: nodes – number of categories, activation - “softamax”

All layers use a starting weight initializer – Glorot normal

Glorot normal initializer – mean of 0, stddev = sqrt(2 / (fan_in + fan_out))

where fan_in is the number of input units in the weight tensor and fan_out is the number of output units in the weight tensor.(Keras documentation)

The optimizer - “Adam”

Loss function – categorical_crossentropy

Total number of trainable parameters – 167 302.

Benchmark

A simple benchmark model is a random choice of a category.

The accuracy of a random choice is expected to be 1 over the number of categories.

In this project benchmark accuracy is 1/6 = 0.167

III. Methodology

Data Preprocessing

The preprocessing that is done before feeding to the model:

Scale – originally the size of images were 1024 by 1024, it was scaled to 0.25 of the original size to 256 by 256. This was done to reduce the number of trainable parameters without loosing too much information. The pictures after scaling are still recognizable.

The preprocessing of pictures that is not done before feeding to the model:

Normalization – to make each pixel color value between 0 and 1. This preprocessing step was implemented and tested, but after a few training sessions the model would not start to improve on accuracy and loss. So feeding of raw pixel values was tested and model quickly started to improve accuracy and loss. So normalization was not done.

Implementation

All programing was done using Python programing language.

Libraries used were Keras, Numpy, Pandas, Scikit-learn, Scikit-image.

Implementation in steps:

1. A function was written that would return list of all picture paths and their labels.
2. Labels encoded to one-hot
3. Train-test split (test_size = 0.1) of picture paths and labels, and stored in CSV files.
4. An iterator class was written that would read and preprocess pictures, and return batches of Numpy arrays and their labels, all data could not be taken to memory.
5. Convolutional neural network function was implemented in Keras, that would return a model.
6. Trainer function, that would save weights after each epoch.
7. Tester function, test accuracy on a test pictures.

Refinement

A model could be deeper to achieve higher accuracy, but that would take more training time and resources.

IV. Results

Model Evaluation and Validation

Many parameters were trained and tested, model described above was chosen, because of small number of trainable parameters and reasonably high accuracy.

The model after training of about 100 epochs. Was tested on a test set.

The accuracy of the model on a test set is 0.9567

Model may not be very robust on unseen images, because the images, of the same category, used in training were quite similar, even though the attempt was made to make them as much different from one another. Test set was quite similar to the training set.

This project is a “proof of concept”, so a model that would be used in production should be much bigger so that it would have more categories and be deeper to capture the complexities.

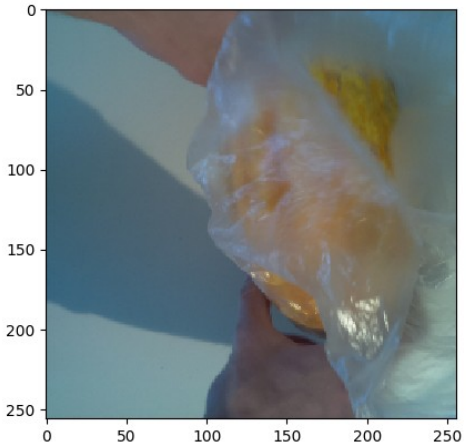
Justification

The model clearly achieved much higher accuracy than the random choice.

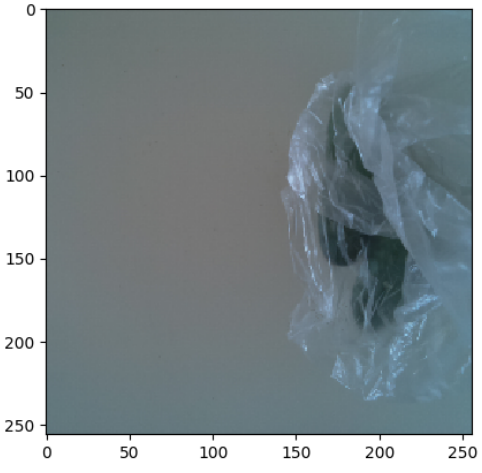
0.9567 and 0.167

Examples of classification

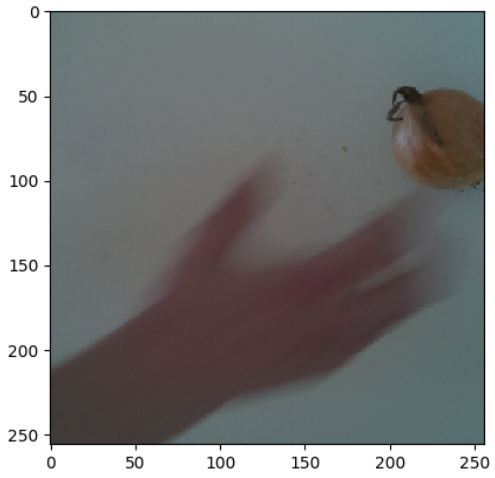
Accurate pumpkin



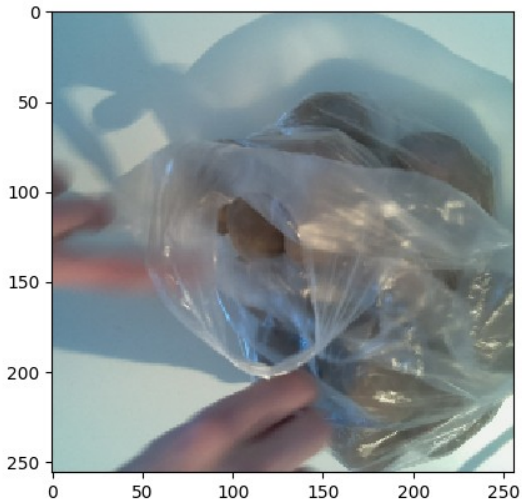
Accurate cucumber



Accurate onion

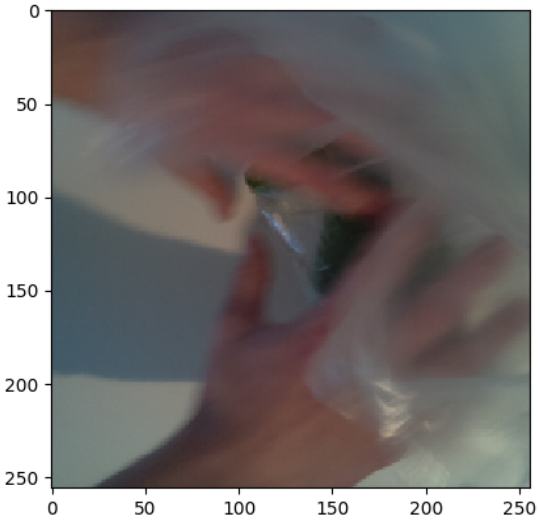


Accurate potatoes



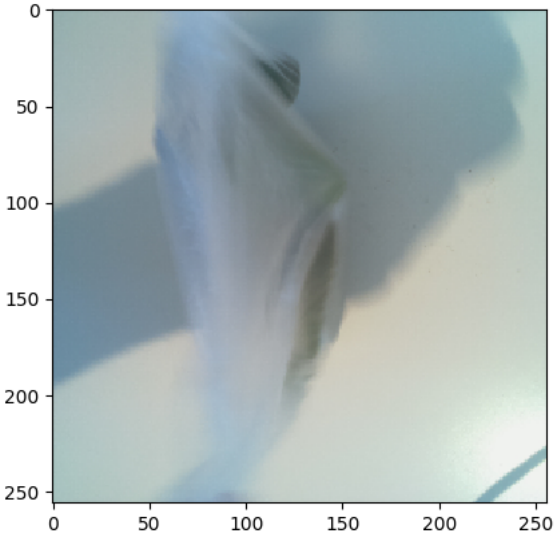
Not accurate should be cucumber

Classified as onion



Not accurate should be cucumber

Classified as potatoes



V. Conclusion

The model showed great results exceeding expectations. However the dataset was only 6 categories and the variability of images in the same category were low.

As a “proof of concept” project was successful, showing that classifying groceries in a plastic bag could be achieved.

A production model should be much larger and deeper.

References

Keras documentation: https://keras.io/