# 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.

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.

The solution tested to this image classification problem is machine learning model – convolutional neural network.

### 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.

I choose this metric because it is simple and easy to understand, and provides the insight how a model would perform in a real world, the only thing that matters whether the model classifies correctly or not.

# 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.






Pictures were taken at home with camera placed above, capturing an image every second. The items in pictures do not vary a great deal, but an attempt was made to introduce variability to the dataset.

The items were constantly moved during photographing. Also the items were photographed by one, like the lemon in picture above, or in a groups of items, like potatoes pictured above. Items were placed in a bag and photographed like most of the images above, mimicking the real world. In addition many images contain hands, again like it would likely be in the real world.

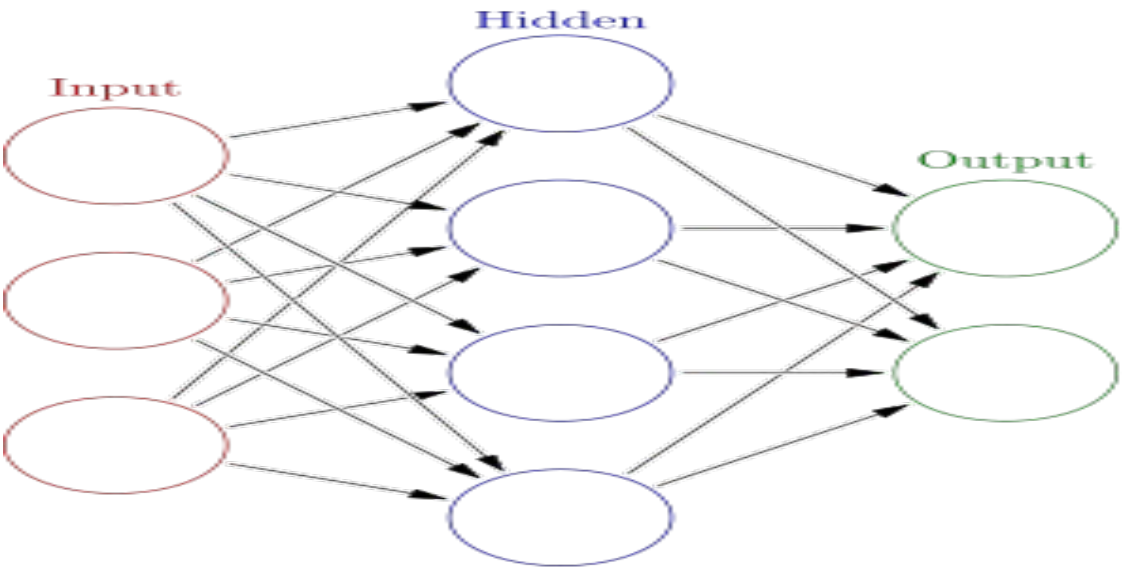
## Algorithms and Techniques

A convolutional neural network was used.

Neural network – network of artificial “neurons”,

Artificial “neuron” - inputs are multiplied by their respective weights, summed and passed through a function.

A network could be deep. Deep neural network means that output of one layer of “neurons” are inputs of other layer.




Convolutional network – is a network that has layer of shared weights. Easy way to understand a convolutional layer is to think of a small neural net that passes through the input space and creates a feature map of the inputs, this feature map is fed to following layer of the network. The advantage is that much less trainable weights are used in a convolution than in a fully connected layer.

Network is optimized by changing weights in each “neuron”.

Neural network approach was chosen because it could effectively adapt to the training set, while being resilient to noise.

A convolutional network model was chosen, because it could classify images regardless of their location in the image, also it would require much less trainable parameters than an equivalent fully connected neural network.

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 little parameters to train, but also retain a good accuracy score.

### 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.

The architecture of the convolutional neural network 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 - “softmax”

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.

### Refinement

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

What could be altered to possibly improve accuracy is the number of convolutional layers. To achieve that strides in existing convolutional and max pooling layers should be lowered, so that the inputs to other layers would not shrink so much. This may improve accuracy by capturing more complexities of the items classified.

## 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

The model may be robust enough for this problem, because there are only 6 categories and these are quite distinct, some have different colors. Model usually fail to accurately classify the images that would be hard even for a human, for example pictures where only plastic bag is recognizable. However all pictures have a white background, so most probably if a picture with a different background is fed to the model, it would fail to classify correctly.

Pictures in the test set were quite similar to the training set. So to firmly state that the model is robust, more variability in the train and test set should be introduced.

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.

Overall model was quite successful, showing that a convolutional net could quite accurately classify items, regardless whether they are in a bag or not.

## Justification

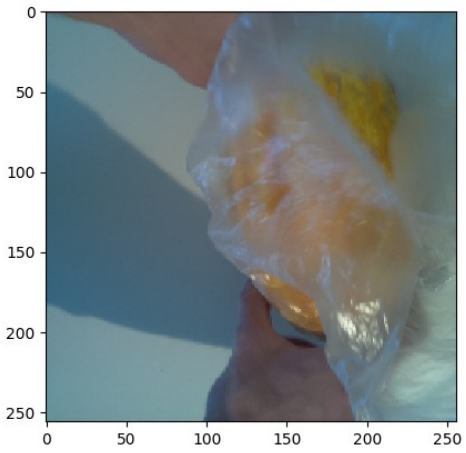
The model clearly achieved much higher accuracy than the random choice.

0.9567 against 0.167

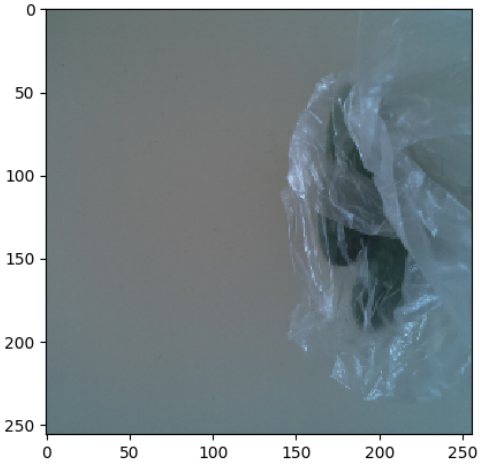
## **Free-form visualization**

The visualization shows what type of images model classifies correctly and where the model falls short.

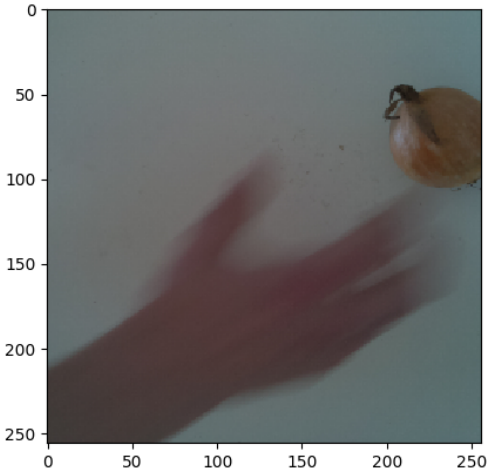
Accurate pumpkin



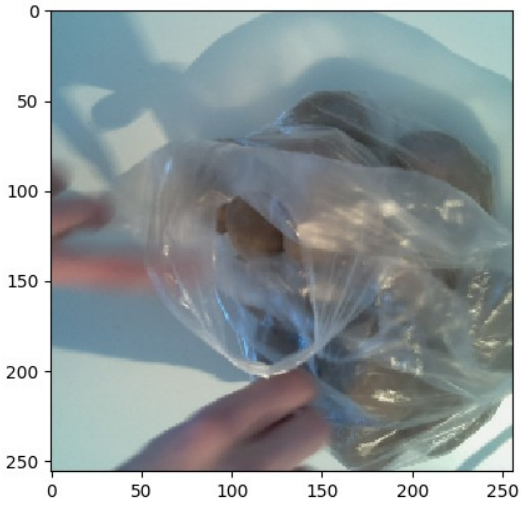

Accurate cucumber




Accurate onion

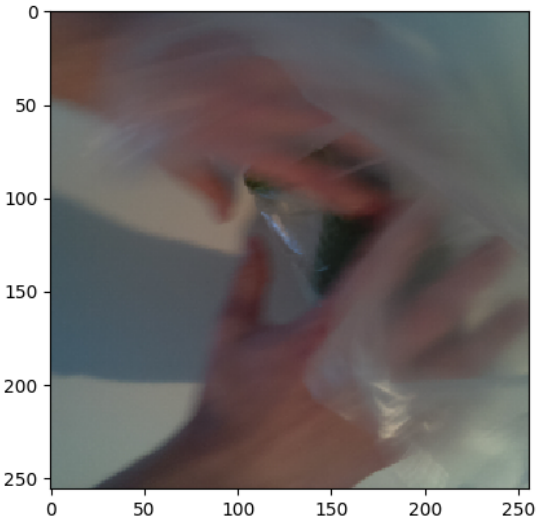



Accurate potatoes



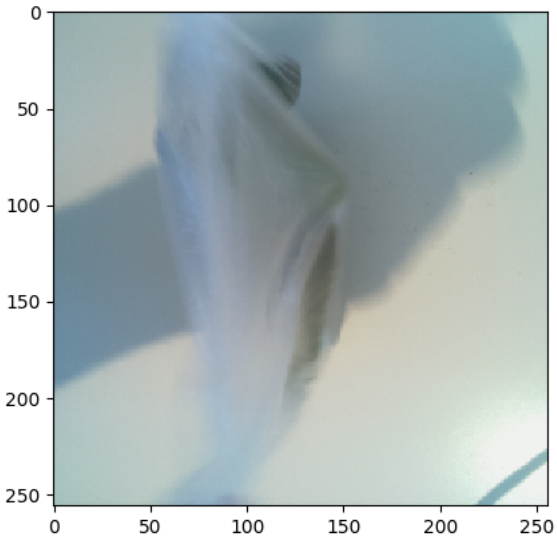
Not accurate – should be cucumber

Classified as onion




Not accurate – should be cucumber

Classified as potatoes




# V. Conclusion

A “proof of concept” Image classification problem of common fruit and vegetable pictures, was tackled. A convolutional neural network was tested as a solution to the problem. The model showed promising results and accuracy on a test set of about 95%

## Reflection

One interesting aspect found during model training was that a model showed better results when fed raw pixel values (0 to 255). When the pixel values were divided by 255 to make their values from 0 to 1 the model would hardly improve.

## Improvement

A model robustness could be improved if training and testing pictures would be of more varied item.

Model accuracy could be improved by making the model - covolutional neural network deeper, add more convolutional layers and reducing the stride step. But the model should not become too big and overly complex, then it may become hard and long to train. Minimal amount of trainable parameters that achieve desired accuracy is optimal.

Model accuracy also could be improved by longer training times.

## References

Keras documentation: https://keras.io/

Image:https://en.wikipedia.org/wiki/Artificial_neural_network#/media/File:Colored_neural_network.svg