Application of deep learning for fruit recognition using Convolutional Neural Networks and Transfer Learning

Adu Boahene Quarshie
Faculty of Engineering, Environment and Computing
MSc. Data Science and Computational Intelligence
Coventry University, Coventry. United Kingdom
quarshiea@uni.coventry.ac.uk

Abstract: ***In this paper, we experiment with a fruit dataset from Kaggle.com using deep learning techniques. We compare a vanilla model to a pretrained model using Transfer Learning training a pretrained model and a vanilla model and a CNN architecture from scratch. We discuss the algorithms, experimentation results and the model that best fit for image recognition and prediction for the dataset. The programming language used is python and it utilises frameworks like TensorFlow.***
An evaluation of the three techniques are carried out to discuss the cost, effectiveness, advantages and disadvantages of each to inform our choice for the fruit recognition.

Keywords – Deep Learning; Artificial Neural Network; Convolutional Neural Networks; Vanilla Model; Fruits; Object Recognition; Transfer Learning; Image Processing; Computer Vision.

I. INTRODUCTION

This year, Tesla announced it was acquiring a computer vision start-up which would boost its development of autonomous driving greatly. When it comes to industries using various applications of computer vision and augmented reality, neural networks are essential in making that possible today through image recognition. Today, we see image recognition used in so many things that it is easy to overlook the power of deep learning in our everyday devices.

Google released a beta version of it's 'Live View' in Google Maps. This feature lets you scan your environment and by doing that, it'll give you an augmented assistance to your location whenever you choose to walk to a destination entered in the app. Google has been using deep learning for so many things including a live translator through your mobile camera where you can point your camera at a sign in a different language and have it translated to your preferred language.

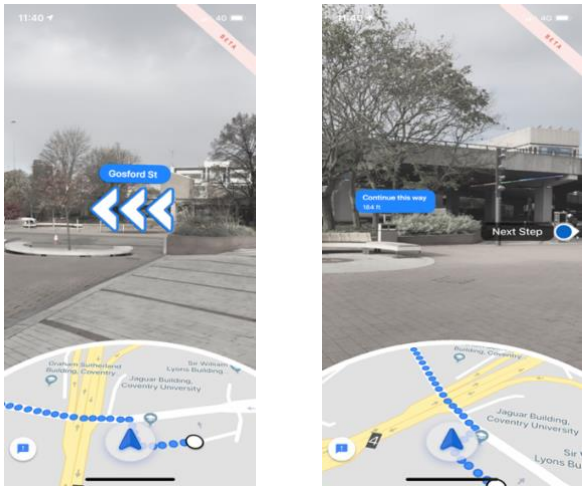

FIG 1. IMAGES FROM GOOGLE MAPS LIVE VIEW (BETA)

Google is not the only company using deep learning in the everyday services. Companies like Apple use deep learning for a lot of practical services and some goes as simple as segmenting already taken pictures into categories, for users to easily finding pictures by entering a keyword.

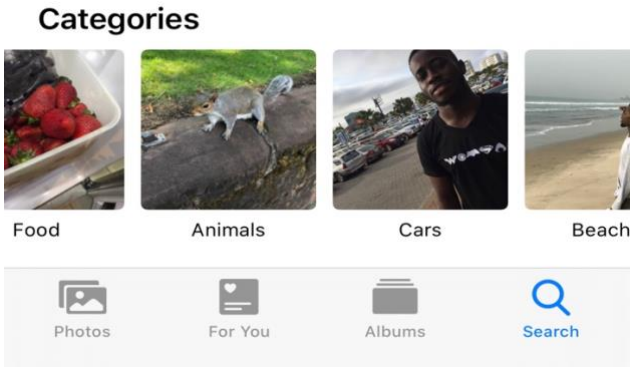

FIG 2. IMAGE OF PHOTOS APP ON AN IPHONE SHOWING CATEGORIES

Amazon opened up a new grocery store called Amazon go. This was the world's first grocery store where you didn't have to have any human interaction whatsoever with your shopping. How the store worked was using a lot of cameras and deep learning to register any item picked from their shelves into your basket and it will automatically charge your account or card with the items you walk out of the store with. This and many other things are some of the feats of deep learning in today's technology and it all starts from object recognition.

In this project, we focus on fruits recognition and we use a dataset with 120 classes of fruits. We use this dataset because when it comes to fruit recognition, there are many applications for it. There is the application of deep learning in fruit recognition for smart fridges that can tell its users the items they're short of, there is also its uses in autonomous fruit harvesting, there are also the application of neural networks to detect ripe and unripe fruits, etc.
In this dataset, there are different species of fruits which if we train our neural network to give a high accuracy of recognising, it'll be effective since most fruits of the same species have the same look and it can be hard to tell it apart. When you take apples for example, in the dataset we have these types of apple varieties: Crimson Snow, Golden, Golden-Red, Granny Smith, Pink Lady, Red and Red Delicious.

When it comes to object recognition, the background of the images used to train the models are important, there are different variances when it comes to fruits because there are always lighting conditions when it comes to the images used to train the models, the shapes, colours and sizes of the fruits too as they don't always in the same size, colour or shapes. The backgrounds for the dataset

we're using is constant and this helps us to train our models to recognise fruits under very conducive conditions before moving on to algorithms that are more complicated for tasks like identifying fruits in an orchid or garden for harvesting. This report is organised in the following order:

- II. The Dataset
- III. Experimental Setup and Results
- IV. Conclusion and Evaluation
- V. Appendix
- VI. References

II. THE DATASET

It is important that when training a neural network for object recognition, high-quality images are used to get a good classifier. The dataset is obtained from Kaggle.com, it is called Fruit 360 and as of today (11th November 2019) the dataset on Kaggle has a total of 82213 images with one fruit or vegetable per image, these images are split into Training, Test and Multiple fruits test folders. The training set size is 61488, the test set size is 20622 and the Multiple fruits set size is 103 images which is made of different fruits (or fruit class) and vegetables per image. The dataset is made up of 120 classes of fruits and vegetables. The images have a constant background and have a regular image size which is 100 x 100 pixels. Some of the images were rotated to give the images some uniformity. Different varieties of the same fruits or vegetables were stored as belonging to different classes, like apples and bananas, etc.

For how this dataset was made, fruits and vegetables were planted in a shaft of a low-speed motor with a speed of 3 rpm and a short video was recorded with a duration of 20 seconds. The gear used for this recording was a Logitech C920 webcam, a white sheet of paper was placed behind the fruits to be utilised as a background.

After the recording, there were differences in the background in the footage acquired due to the variation in the lighting conditions of the environment. An algorithm was developed which helped extract the fruits and vegetables from the background.

How the algorithm (which was a flood fill type) worked was that it started from each of the edge of the image in the dataset and then marked all pixels in there. Then, we mark all pixels found in the neighbourhood of the already marked pixels for which the distance between colours is less than a prescribed value. The previous process is repeated until no more pixels can be marked.

All the marked pixels are then considered as being background (which is then filled with white) and the rest of the pixels are also considered as belonging to the object which is the fruit or vegetable.

The maximum value for the distance between 2 neighbour pixels is a parameter of the algorithm and is set (by trial and error) for each video.

The pictures which were in the multiple fruit test folder was taken with a Nexus 5X phone.

In every folder in the training set, there are over 400 images and in every folder in the test folder, there are over 100 images for each fruit or vegetable.

III. EXPERIMENTAL SETUP

In this paper, we train models to recognise fruits, we use two methods which is Transfer learning and Convolutional Neural Networks. However, in the Transfer learning, we compare two models which is a vanilla model and a pretrained model which brings the total number of models used to 3. Before we get into each model and how it works for our dataset, let's look into deep learning.

DEEP LEARNING

It is without doubt that when it comes to object recognition, the successes seen are mostly due to artificial neural networks. Deep learning is a subset of machine learning which focuses on learning with layers of incremental representations. The layers that contribute to the model's training is called *depth;* each layer learns to convert its train data into a slightly more composite form. Below is how a deep learning algorithm with multiple layers for recognising images to convert them into text looks like.

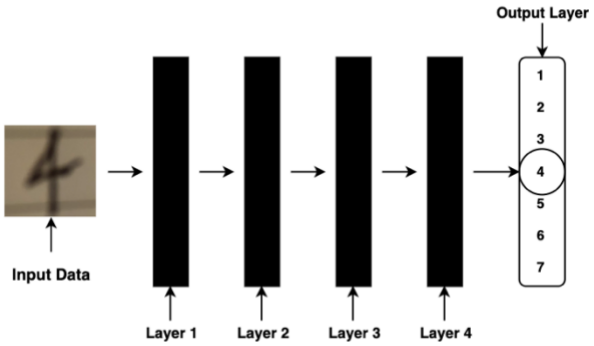

FIG 3. A DEEP NEURAL NETWORK FOR DIGIT CLASSIFICATION

To understand the above, what each layer in the algorithm does is stored in the layer's weight. The weight acts as parameters for the transformation that happens in the layers. The learning method for our dataset and neural networks are mostly supervised. So, with our dataset, the training set are in classes which have labels to help the model learn what fruit or vegetable belongs to a specific class.

TRANSFER LEARNING

Transfer Learning is a machine learning technique which is trained for a specific task and then repurposed as a start for another task. With this technique, a base network is trained first on a base input specific to a task, then its learned features are later transferred to a second target network to be trained on a target dataset and task. There are three possible advantages to using transfer learning in this project. Their advantages are that when using transfer learning, there is a higher start, a higher slope and higher asymptote. A pretrained model and a vanilla model is used in our project to show the comparison between the two learning techniques.

PRETRAINED MODEL AND VANILA MODEL

A pretrained model is a model with its weights and parameters of the network already trained on large amounts of datasets, this model is then fine-tuned to fit our own dataset. With this idea, the pretrained model will inform the initialised parameters which leads to a faster convergence or it will end up acting as a feature extractor that is fixed for a particular task.

A vanilla model or neural network is a network with one hidden layer neural network or multilayer perceptron network. Compared to a pretrained model, this is simple and a lazy learner.

We import new data which is basically pretrained models on Kaggle. The Keras pretrained models contain the following: Xception, VGG16, VGG19, ResNet50, InceptionV3 and InceptionResNetV2.
We import the pretrained models into a cache where keras finds them using:

```
cache_dir = expanduser(join('~', '.keras'))
if not exists(cache_dir):
    makedirs(cache_dir)
models_dir = join(cache_dir, 'models')
if not exists(models_dir):
    makedirs(models_dir)

!cp ../input/keras-pretrained-models/*notop* ~/.keras/models/
!cp ../input/keras-pretrained-models/imagenet_class_index.json ~/.keras/models/
!cp ../input/keras-pretrained-models/resnet50* ~/.keras/models/

print("Available Pretrained Models:\n")
!ls ~/.keras/models
```

BUILDING MODELS

Here, we load the ResNet50 model with ImageNet weights. We then remove the top so we can add our own layer according to the number of our classes in our dataset which is 120 classes. We then add a suitable number of layers to complete the build of the model architecture.

With building the pretrained model, we import the pretrained model with the pretrained weights which does not come with fully connected layers. Before we build the models, we set the dimensions of our images to best fit the pretrained model. The input size for ResNet 50 is 224 by 224 by 3. Our batch size is 64.
With building both models we follow some processes, first we add a global spatial averaging pooling layer. Then, we add a fully connected layer, a fully connected output and create the full network so we can train it using:

```
inception_base = applications.ResNet50(weights='imagenet', include_top=False)

x = inception_base.output
x = GlobalAveragePooling2D()(x)

x = Dense(512, activation='relu')(x)

predictions = Dense(120, activation='softmax')(x)

inception_transfer = Model(inputs=inception_base.input, outputs=predictions)
```

After building the models, we compile the models using:

```
inception_transfer.compile(loss='categorical_crossentropy',
optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
metrics=['accuracy'])

inception_transfer_vanilla.compile(loss='categorical_crossentropy',
optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
metrics=['accuracy'])
```

We use accuracy as our metric when compiling the models. We then move on to train and validate the models using 5 Epochs. We use the fit_generator() function because we are using object of the ImageDataGenerator class to look for the data. Due to computational limitations, we used a small and randomised amount of our dataset to train and validate our model which gave us the following results.

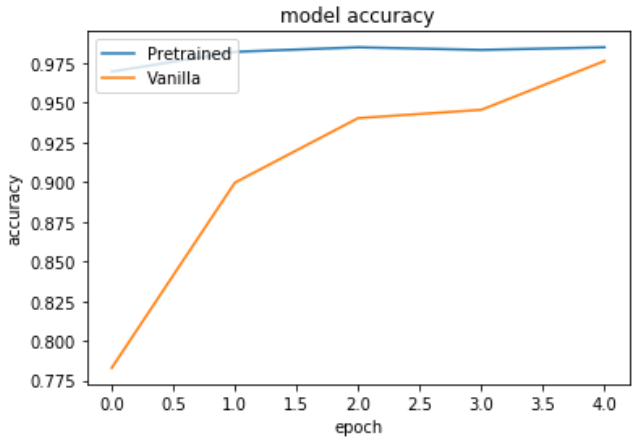

FIG 4. ACCURACY AGAINST EPOCH PLOT FOR BOTH MODELS

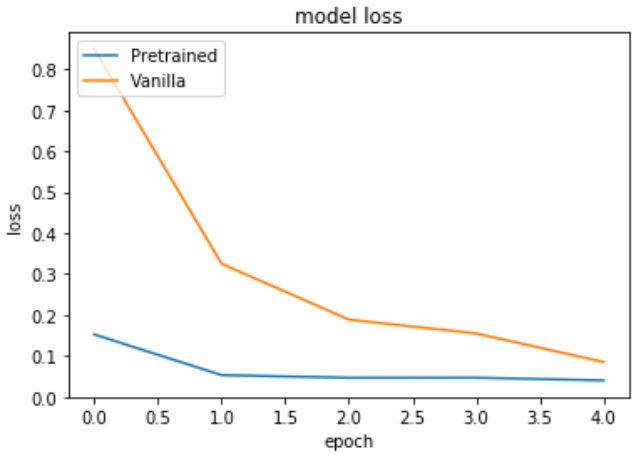

FIG 5. LOSS AGAINST EPOCHS FOR BOTH MODELS

CONVOLUTIONAL NEURAL NETWORKS

This is a deep learning technique that is composed of layers like convolutional layers, pooling layers, RELU layers, fully connected layers and loss layers. In a convolutional neural network architecture, every layer is succeeded a RELU layer which is followed by a pooling layer, one or more convolutional layer and lastly a fully connected layer. A distinction factor about a CNN and a regular model is its ability to take into consideration the structure of the images used during the process. In our fruit recognition task, we build a simple convolutional neural network from scratch, we use 3 convolutional layers followed by maxpooling layers then finally a dropout layer, flatten and some fully connected layers.
The code is below:

```
1   model = Sequential()
2   model.add(Conv2D(filters = 16, kernel_size =
    (3,3),strides=(2,2),input_shape=(224,224,3),pad
    ding='same'))
3   model.add(Activation('relu'))
4   model.add(MaxPooling2D(pool_size=2))

5   model.add(Conv2D(filters = 32,kernel_size =
    2,activation= 'relu',padding='same'))
6   model.add(MaxPooling2D(pool_size=2))

7   model.add(Conv2D(filters = 64,kernel_size =
    2,activation= 'relu',padding='same'))
8   model.add(MaxPooling2D(pool_size=2))

9   model.add(Conv2D(filters = 128,kernel_size =
    2,activation= 'relu',padding='same'))
10  model.add(MaxPooling2D(pool_size=2))

11  model.add(Dropout(0.3))
12  model.add(Flatten())
13  model.add(Dense(200))
14  model.add(Activation('relu'))
15  model.add(Dropout(0.4))
16  model.add(Dense(120,activation = 'softmax'))
17  model.summary()
```

After the model was trained and test on the validation set, an accuracy of 0.9978 was achieved.

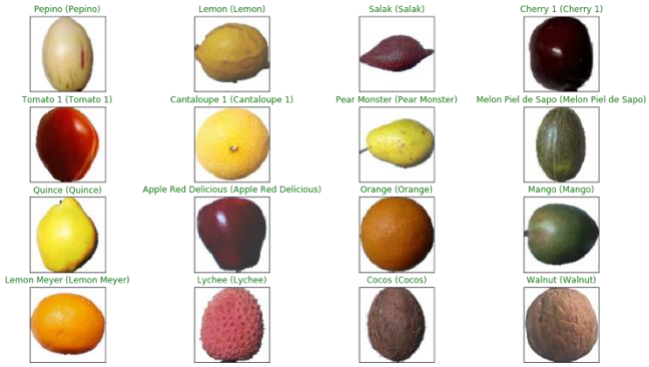

FIG 6. A RANDOM SAMPLE OF TEST IMAGES WITH PREDICTED LABEL AND THEIR GROUND TRUTH

A representation of the accuracy against epochs and loss against epochs follows:

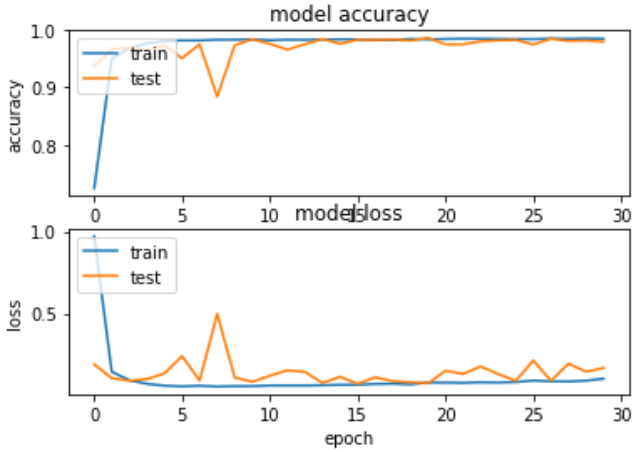

FIG 7. AN ACCURACY AND LOSS GRAPH AGAINST EPOCHS FOR TRAINING AND TESTING SETS

IV. CONCLUSION AND EVALUATION

Considering the accuracies and losses achieved by the three models, The CNN models is the best model that fits our task for fruit recognition achieving an accuracy of 0.9978 after 15 epochs.

VII. REFERENCES

[1] Horea Muresan, Mihai Oltean, Fruit recognition from images using deep learning, Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42, 2018.
[2]. Convolutional Neural Network: How it works? Anindita Pani, Dec 21, 2018.
[3] Muresan, H., and Oltean, M. Fruits 360 dataset on github. [Online; accessed 2.11.2019].
[4] O'Boyle, B., and Hall, C. What is google lens and how do you use it? [Online; accessed 2.11.2019].
[5] TensorFlow. TensorFlow. [Online; accessed 04.11.2019].
[6] Song, Y., Glasbey, C., Horgan, G., Polder, G., Dieleman, J., and van der Heijden, G. Automatic fruit recognition and counting from multiple images. Biosystems Engineering 118 (2014), 203 – 215.
[7] Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., and McCool, C. Deepfruits: A fruit detection system using deep neural networks. Sensors 16, 8 (2016).
[8] Ren, S., He, K., Girshick, R. B., and Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. CoRR abs/1506.01497 (2015).
[9] Puttemans, S., Vanbrabant, Y., Tits, L., and Goedem, T. Automated visual fruit detection for harvest estimation and robotic harvesting. In 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA) (Dec 2016), pp. 1–6.
[10] Kapach, K., Barnea, E., Mairon, R., Edan, Y., and Ben-Shahar, O. Computer vision for fruit harvesting robots – state of the art and challenges ahead. Int. J. Comput. Vision Robot. 3, 1/2 (Apr. 2012), 4–34.
[11] Selvaraj, A., Shebiah, N., Nidhyananthan, S., and Ganesan, L. Fruit recognition using colour and texture features. Journal of Emerging Trends in Computing and Information Sciences 1 (10 2010), 90–94.
[12] François Chollet, Deep Learning with Python, 2018, Manning publications Co. Page 4 – 12.

VII. APPENDIX

The codes are in a google drive:
https://drive.google.com/drive/folders/1ExA35nue4ihI9p75wouYwebFB-qdMq7N?usp=sharing