CERTIFICATE

Certified that this project report entitiled "**Classification of Fruits using Convolutional Neural Networks**" submitted by

"SAYAN ROY"

a student of Information Technology Department, Gauhati University, Guwahati, Assam who carried on the project under my supervision.

This report has not been submitted by any other university or institution for the award of degree.

SIGNATURE	SIGNATURE
(HEAD OF DEPARTMENT)	SUPERVISOR

DECLARATION

I hereby declare that:

a) The work contained in this report has been done by me under the guidance of my supervisor.
b) The work has not been submitted to any institute for any degree or diploma.
c) I have followed the guidelines provided by the institute in preparing the report.
d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other source, I have given the credit to them by citing them in the text of the thesis and giving their details in the reference.

Signature:

Date:

Name: Sayan Roy

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my guide Dr. Surya Prakash Sir of Computer Science and Engineering Dept,IIT Indore for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by Akhilesh Mohan Srivastava Sir time to time shall carry me a long way in the journey of my life on which I am about to embark.

I am highly indebted to Computer Science and Engineering dept. of IIT Indore for the constant guidance and supervision as well as providing necessary information regarding the project and also for their support in completing the project.

Lastly, I thank Almighty, my parents, my brother and my friends for their constant support and encouragement without which this project would not have been possible.

ABSTRACT

This study proposes the development of a convolutional neural network to classify fifteen kinds of fruits, which in addition to the classifier, would inform about the accuracy of the results obtained. In the case of a low value of the accuracy, it would inform the user regarding the most suitable species (varieties) of products.

INTRODUCTION

The notation of recognition (identification, classification) can also be understood in different ways:

- As the recognition of a fruit (distinguishing a fruit from another object, e.g., a leaf, a background),
- Recognizing the species of a fruit (e.g., apple from a pear)
- Recognizing a variety of a given species of fruit (e.g., Golden Delicious apples from brae-burn apple).

In the case of retail systems, the last two applications have special significance. An examination of the literature suggests that the effectiveness of fruit and vegetable classification using various machine learning methods (support vector machine, k-nearest neighbor, decision trees, neural networks), especially recent advancements in deep learning, is great. However, the construction of online fruit and vegetable classification systems in retail sales is challenged by the required model learning time and promptness in receiving the classification result, as well as the accuracy of the model prediction. In the case of complex, multi-layered models of deep neural networks, the learning and inference time can be significant. Therefore, in the analyzed application, the most preferred models are those that provide a solution relatively quickly and with high classification accuracy.

Classification of fruits and vegetables is a relatively complex problem owing to the huge number of varieties. Considerable differences in appearance exist within species and varieties, including irregular shapes, colors, and textures. Furthermore, images range widely in lightning conditions, distance, and angle of the camera; all of which result in distorted images. Another problem is the partial or full occlusion of the object. These constraints have led to the lack of multi-class automated fruit and vegetable classification systems in real-life applications.

The primary aim of this project is to propose a method for fruit classification, which in addition to the classifier, would inform about the accuracy of the results obtained. In the case of a low value of the accuracy, it would inform the user regarding the most suitable species (varieties) of products. This work focuses on the classification of 15 species of popular fruits using a convolutional neural network (CNN).

PROBLEM STATEMENT

One of the important quality features of fruits is its appearance. Recognizing different kinds of fruits and vegetables is perhaps the most difficult task in supermarkets and fruit shops. Retail sales systems based on bar code identification require the seller (cashier) to enter the unique code of the given fruit or vegetable because they are individually sold by weight. This procedure often leads to mistakes because the seller must correctly recognize every type of vegetable and fruit; a significant challenge even for highly-trained employees. A partial solution to this problem is the introduction of an inventory with photos and codes. Unfortunately, this requires the cashier to browse the catalog during check-out, extending the time of the transaction. In the case of self-service sales, the species types and varieties of fruits must be specified by the buyer. Unsurprisingly, this can often result in the misidentification of fruits by buyers (e.g., Conference pear instead of Bartlett pear). Independent indication of the product, in addition to both honest and deliberate mistakes (purposeful indication of a less expensive species/variety of fruit/vegetable) can lead to business losses. The likelihood of an incorrect assessment increases when different fresh products are mixed up. One potential solution to this challenge is the automatic recognition of the fruits and vegetables. An overview of the process of building a convolutional neural network to classify 15 kinds of fruits has been presented in this report.

DATASET DESCRIPTION

In the proposed project, I have taken a set of 15 kinds of fruits from the *Kaggle Fruits 360 dataset*.[1] It’s a high-quality dataset of images containing fruits and vegetables. A total of 120 fruits and vegetables are included.

Total number of images: 82213.
Training set size: 61488 images (one fruit or vegetable per image).
Test set size: 20622 images (one fruit or vegetable per image).
Multi-fruits set size: 103 images (more than one fruit (or fruit class) per image)
Number of classes: 120 (fruits and vegetables).
Image size: 100x100 pixels.
Filename format: image_index_100.jpg (e.g. 32_100.jpg, “100” comes from the image size.)

For this particular project, I have taken a set of 15 kinds of fruits from the dataset.

Fruit	**No. of Training images**	**No. of testing images**
Apple Braeburn	492	164
Apricot	492	164
Avocado	427	143
Banana	490	166
Clementine	490	166
Cocos	490	166
Lemon	490	166
Limes	492	164
Mandarin	479	160
Orange	492	164
Peach	490	166
Pineapple	447	151
Plum	492	164
Pomegranate	490	166
Raspberry	492	164

REQUIREMENTS

Energy efficiency of the underlying memory systems is a huge issue in most neural network accelerator designs. It is imperative to understand the characteristics and behavior of data in the algorithm of neural networks to gain an insight into their impact on memory-systems [9]. The current work studies such impacts of convolutional neural networks on the energy efficiency of memory-systems by analyzing the structuring of datasets in convolutional neural networks, their memory space and bandwidth requirements, the energy efficiency of their accesses in DDR memory systems versus 3D-stacked DRAM systems and the future direction of this area. Also, one possible architectural implementation using the near-data processing capabilities of highly parallel heterogeneous 3D-stacked DRAM chips of Micron's Hybrid Memory Cube is demonstrated which shows an improvement of above 90% in energy efficiency for acceleration of convolutional neural networks. [2]

EXPERIMENT

The experiment workflow would go on like this:

- Defining the layers
- Compiling the model
- Training the model
- Evaluating the model

Defining the layers:

The easy layers to figure out are the input and output layers. The input layer has one input node and in Keras there is not need to define it. The output layer depends on the type of neural network. If it's a regressor or a binary classifier, then it has one output —we are either returning a number or a true/false value. If it's a multi-class classifier with a softmax function at the end, you will need one node per class —in our case, it's the latter.

```
  model = Sequential()#input
  model.add(Dense(units = len(trained_classes))) #output
```

For the convolution layers, we need to strike a balance between performance and quality of the output. In this particular case, I noticed that having a single convolution already produced 97% accuracy, and I added an extra convolution because the convergence of the validation accuracy seemed smoother. Since all the images were of high quality —a rotation of a single fruit with transparent background— a single convolution would've done the job. I bet that there are cases with lower-quality images in which the extra convolutions will improve the performance of the network.

```
#First Convolution
model.add(Conv2D(filters=64, kernel_size=(3,3), padding = 'same', strides=(1, 
1), input_shape = (im_size, im_size, channels), name='conv2d_1'))
#second convolution
  model.add(Conv2D(filters = 128, kernel_size = (3,3), padding = 'same', 
name='conv2d_2'))
```

At the end of all the convolutions, we insert a pooling layer which down-samples the feature maps and reduces the dimensionality of the problem. Pooling takes the maximum value of a sliding receptive field over each feature map. It also helps make the detection of certain features invariant to scale and orientation. In other words, it makes the network more robust by focusing on the most important features of the image.

```
model.add(MaxPooling2D(pool_size = (2,2)))
```

After the pooling layer, we flatten the resulting matrix into a vector and we feed it through a fully-connected network. Each element in this vector will correspond to one or more features detected in the convolution, and the purpose of this fully-connected layers is to make classifications. In other words, after the convolution, we staple a standard neural network classifier. At the tail end, the fully-connected network will produce a probability distribution per each class, and these distributions will depend on the weight attributed to each neuron in the final classification. Though back propagation, Keras will automatically adjust the weight of each of these connections in order to minimize the classification error. [4]

```
model.add(Flatten())
model.add(Dense(units = 250, name='dense_1' ) )
model.add(Activation('relu', name='activation_dense_1'))

  # output layer with the number of units equal to the number of categories
model.add(Dense(units = len(trained_classes_labels), name='dense_2'))
model.add(Activation('softmax', name='activation_final'))
```

Compiling the model

After we define our network architecture, we can compile the model using *model.compile()*. For the loss function, I used categorical-cross-entropy because this is a multi-class classification problem. Binary-cross-entropy would be the one in a binary classification problem and mean-squared-error for a regression. Regarding the optimizer, RMSprop is the recommended one in the Keras documentation. [10]

Fitting the model

This is the step which actually generates the model. There are two functions which you can use to fit a model in Keras.

- **fit** loads the whole dataset in memory
- **fit_generator** feeds off a generator and it's intended for large datasets

I have used fit_generator in the project because I'm using an iterator (flow_from_directory) to load the images.

The fit process runs a full cycle of training and testing multiple times. Each of this runs is called an **epoch**. After each epoch we can measure the accuracy and loss of the model. If we run a backward propagation algorithm, we can minimize the error of the model using gradient descent on each successive epoch. The number of samples used for the gradient descent calculation is specified with the batch_size parameter. Keras automatically runs the back propagation algorithm for you and adjusts the weights. After each run, you can instruct Keras to run certain commands, like for example saving a model checkpoint with the current weights.

For computational simplicity, not all training data is fed to the network at once. Rather, let's say we have total 1600 images, we divide them in small batches say of size 16 or 32 called batch-size. Hence, it will take 100 or 50 rounds(iterations) for complete data to be used for training. This is called one epoch, i.e. in one epoch the networks sees all the training images once.

The fit_generator returns a history object with 4 measures: training loss, training accuracy, validation loss and validation accuracy. The training metrics are a comparison of the output of the model with the training images and the validation metrics are a comparison with the test or validation images (the ones not used for training).

Since we want to know how good the model is generalizing, the focus should be on the validation metrics. Each epoch constitutes a complete training and validation cycle and Keras provides an automated way of saving the best results of all the epochs with the ModelCheckpoint callback. [4]

Evaluating the model

I followed the Keras documentation on model visualization to generate the accuracy and loss plots. Tensorboard can also be used, which offers an easy way to compare multiple runs with different parameters. Plotting is very useful as it allows us to see whether the model is undefitting or overfitting. Underfitting, or under-learning is pretty easy to spot: the training lines don't converge as expected (to 0 loss or 100% accuracy). Overfitting is a little bit trickier. For such a clean dataset, any sudden change in the validation line is a sign of overfitting. [5] If the accuracy stops improving is also a sign of overfitting, this is when the early stopping feature might become handy.

Classification accuracy alone can be misleading if we have an unequal number of observations in each class or if you have more than two classes in your dataset. One may get a classification accuracy of 90%, but he/she won't know if holds throughout because all classes are being predicted equally well or whether one or two classes are being neglected by the model. Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making[6][7]. A confusion matrix is a technique for summarizing the performance of a classification algorithm. The number of correct and incorrect predictions are summarized with count values and broken down by each class. It is this breakdown that overcomes the limitation of using classification accuracy alone. [8]

FUTURE SCOPE

- The future direction is to test the method using a larger dataset containing greater amounts of different fruit and vegetable varieties of different species.
- Classification of fruits and vegetables is a relatively complex problem owing to the huge number of varieties. Considerable differences in appearance exist within species and varieties, including irregular shapes, colors, and textures. Furthermore, images range widely in lightning conditions, distance, and angle of the camera; all of which result in distorted images. These factors should be taken care of while building the succeeding model.
- It is also preferable to build a fruit and vegetable dataset with more demanding images, which will ultimately be the true test of the system.
- An interesting research direction will be testing the system with a dataset containing images of fruits and vegetables wrapped in a transparent plastic bag. This situation may cause uncertainty of the obtained result.
- Another hitch is the partial or full occlusion of the object. These constraints have led to the lack of multi-class automated fruit and vegetable classification systems in real-life applications.

CONCLUSION

This report has discussed the development of a convolutional neural network to classify fifteen kinds of fruits. The objectives of this project were to develop the necessary model to identify the type of the fruit. The validation using 2270 images of the fifteen varieties of fruits indicated that classification accuracy for this method was excellent (~97%). By this deep learning algorithm, the model was able to identify almost every fruit sample. Thus, Neural Networks can be pre-dominantly used for the classification of fruits as they result in a great degree of accuracy. We conclude that Neural Networks classifiers are a great choice for fruit recognition as it yields good accuracy.

This project introduced us to the concepts of neural networks and the various ways to implement them in real life. Apart from classifying fruits, neural networks can also be used in predicting weather, solving complex problems, recognizing pattern, matching images, face recognition, and many others. It is also important to recognize that the proposed method also had limitations and that should be kept in mind while building the succeeding model.

REFERENCES

[1] Oltean, M. (2019, November 5). *Fruits 360.* Retrieved from kaggle.com/moltean/fruits

[2] Katz M., (2019, January 23). *Memory Requirements for Convolutional Neural Network Hardware Accelerators*. Retrieved from ieeexplore.ieee.org/document/8573527

[3] Saha S., (2018, December 8). *A Comprehensive Guide to Convolutional Neural Networks* – the EL15 way. Retrieved from towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[4] Li, S., (2017, December 4). *Solving A Simple Classification Problem with Python — Fruits' Edition.* Retrieved from towardsdatascience.com/solving-a-simple-classification-problem-with-python-fruits-lovers-edition-d20ab6b071d2

[5] Pradilla, D. 2019 a. *Classifying fruits with a Convolutional Neural Network in Keras*, GitHub, github: danielpradilla/python-keras-fruits

[6] Akilos, R. 2015 a. *Confusion Matrix with Keras flow_from_directory*, GitHub, github: RyanAkilos/3808c17f79e77c4117de35aa68447045

[7] Doring, M., (2018, December 4). *Performance Measures for Multi-Class Problems.* Retrieved from datascienceblog.net/post/machine-learning/performance-measures-multi-class-problems

[8] Wulf, A., McKee S. A., (1995, March). *Hitting the memory wall: implications of the obvious.* Retrieved from dl.acm.org/doi/10.1145/216585.216588

[9] Brownie J., (2020, January 12). *What is a Confusion Matrix in Machine Learning*. Retrieved from machinelearningmastery.com/confusion-matrix-machine-learning

[10] Pradilla D., (2019, February 19). *Classifying fruits with a Convolutional Neural Network in Keras*. Retrieved from towardsdatascience.com/classifying-convolutional-neural-network-keras/

[11] Hunter, J.D., *Matplotlib: A 2D Graphics Environment*, Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

[12] Chollet, F., 2015 a. *keras*, GitHub, github: fchollet/keras

[13] Pedregosa et al., (2011). *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830