Capstone Project

Data Scientist Nanodegree

Shihab Al Nasiri

May 20th, 2022

Almond Classifier

Project Overview

Despite the fact that food industry may consists of various type of businesses like agriculture, food processing and marketing, and that its value is unquestionably important in rapidly increasing population around the globe, it is still surprising that now days the use of manual evaluation by trained workers is still widely used. Additionally, manual evaluation is subjective and prone to human error with inherently excessive costs. Which leads to the importance of utilising existing techniques that will not only add efficiency but also increase the quality of the evaluation. Image processing techniques and pattern recognition can be part of this project endeavour. The application of machine learning has grown over the years and been widely used in the automation of food inspection.

In this project, a flask app was created to classify almond uploaded images where convolution neural networks (CNN) were trained. Model parameters were saved to be used to predict image types. For the purpose of simplicity two categories were initially created for almond types, normal and broken, thus the training and evaluation data was limited and collected for those two types. Nevertheless, the data used was previously created and used for a previous university project and will be reused in this project.

Problem Statement

Workers working in food industry are not always attentive to trivial details, therefore, the necessity of image processing techniques that can process objects at high speed with more efficiency, rises. In this project almond samples for two types – broken and normal – were taken to train a neural network model. The neural networks model architecture called "convolutional" networks short for "CNN" which works best with images. CNN works well with images as they look for pattern at the pixel level then proceed to gather more information for larger areas and more groups of pixels to extract more complex features from images. The trained model will then be used as a backend for the almond classifier app that will predict the weight of the uploaded image and attempt to classify its category.

Metrics

As mentioned above, the goal for this project is to classify two categories for almonds -either normal or broken-, therefore binary crossentropy will be used as loss function. Binary crossentropy is a loss function that is used in binary classification tasks. These are tasks that answer question with only two choices gives either yes or no.

$$\text{Loss} = -\frac{1}{\substack{\text{output}\\\text{size}}} \sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Figure 1Loss function equation

- Where ŷ is the probability of class 1 and 1- Y_i is the probability for class 0.

When the observation belongs to class 1 the first part of the formula become active, and the second part vanishes and vice versa in the case observation's actual class are 0. **Sigmoid** is the only activation function compatible with binary crossentropy.

Accuracy metrics from karas was used to evaluate predictions. Where accuracy calculates the percentage of predicted values that matches with actual values. For instance, if the predicted value is equal to the actual value, then it is considered accurate.

Analysis

Dataset & inputs

Data set for almond was gathered and labelled for a previous project from university and will be reused for this project. As the almond type can be divided into normal, broken, stain, shrivel and mould, in this project the focus was mainly on normal and broken almonds. Therefore, the data set was containing the only two types with the initial split of 20 images for training and validation.

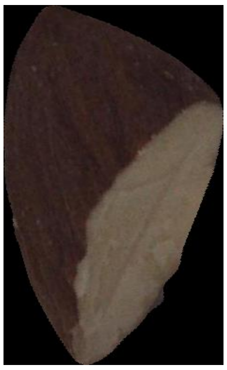


Figure 2 : Broken almond

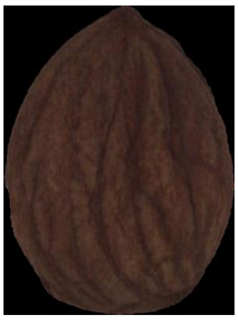


Figure 3 : Normal almond

As it is essential for the data set for training to be of the same size, the number of images collected and used for each class were of the same size in both training.

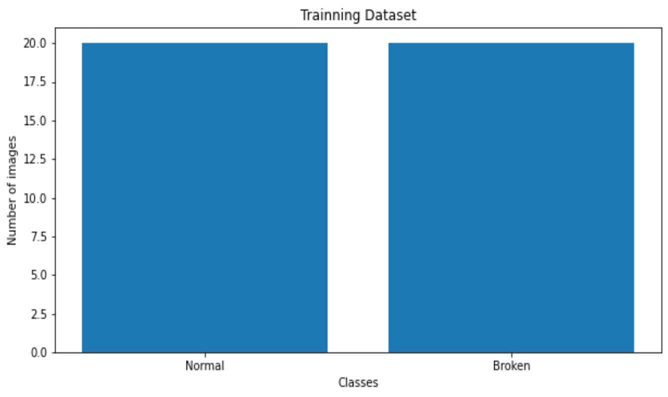
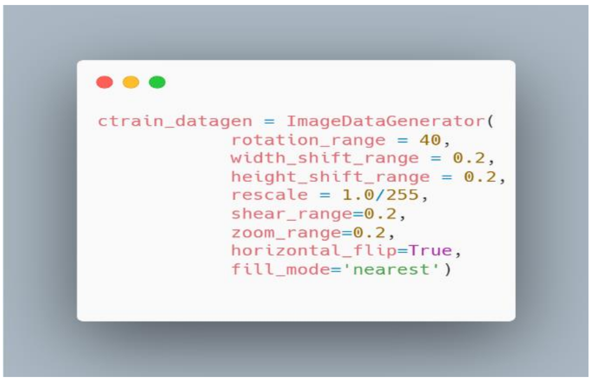


Figure 4 : Dataset

Image data generator from keras will be used to augment the data for training and testing and artificially expand the size of the dataset. Data augmentation will involve transforming images into different version of the same image. In this project, data will be transformed in operation like rotation, shifting, zooming and rescaling.



```
ctrain_datagen = ImageDataGenerator(
            rotation_range = 40,
            width_shift_range = 0.2,
            height_shift_range = 0.2,
            rescale = 1.0/255,
            shear_range=0.2,
            zoom_range=0.2,
            horizontal_flip=True,
            fill_mode='nearest')
```

Figure 5 image augmentation code

Algorithms and Techniques

For classifying broken and normal almonds, a convolutional neural network will be used (CNN) to differentiate between two categories. As this project involves two types of almonds the problem will be binary classification, thus the model will be designed accordingly. Keras framework will be used for importing neural networks and help in creating the convolutional layer's structure. After training the model, the model weights will be used to predict the uploaded image through flask app and attempt classifying its class.

In order to feed the image to the model, the image will be resized using numpy array and transform it size to (150,150) as the model was trained for such inputs. After loading the model functionalities (graph, session, and model), the model will be fed with resized image and then the weight predicted from the model will be interpreted into two classes. Meaning if the weight was less than 0.5 it is considered broken almond, otherwise it is normal almond.

Benchmark

For the benchmark model, the model initially designed was scoring around 50-60% accuracy with one CNN layer and 10 images for training and validation. Which was a good indicator that the model was working with all custom functions used with no overfitting. The goal was to raise the accuracy to more than 80% using more layers and to utilize data augmentation to fight overfitting that comes with complex models.

Methodology

Data Preprocessing

All images are resized to image width of 150 and image height of 150 before fitting to the model. Data augmentations were then applied to all training, testing and validation where operation like flipping, zooming and rotating were applied. With the images having RGB between 0-255 coefficients which will be high for the model to process, thus rescaling was essential to target values between 0 and 1 by rescaling by 1. /255 factor.
As the model was trained for input shape of (150,150,3), it was essential to transform and resize uploaded images before feeding them into prediction which was achieved using a custom function.

Implementation

Using Keras framework, the model created, consisted of 2 convolutional layers followed by ReLU activation and max-pooling layers. Also, another fully connected layers were added on top of the convolutional layers. At the end another single unit and a sigmoid activation function were added which is suitable for binary classification. This was giving a validation accuracy of around 0.6-0.7 after 10 epochs and 5 batch(which was picked arbitrarily).

Then comes the app development where the model was saved and loaded in a flask app backend. A session was then created to use the model to predict uploaded image. The uploaded image name will be saved in a database using SQL lite and the files in a designated folder. When the image is classified and its weight is determined, the class – either normal or broken- and the image will be sent to a front-end web page where the result will be displayed along with it is predicted weight.

Refinement

Though the trained model exceeded the benchmark accuracy, it is believed that there is room for improvement by adding an extra convolutional layer and a drop out. Although the model seems small and uses aggressive dropout, it is believed that it is not overfitting. For the refinement, the number of epochs was increased from 10 to 50 epochs and the training and validation data set increased to twenty images. Which had showed significant change in accuracy and validation accuracy. The recorded accuracy was up to 95% which has exceeded the initial target of the benchmark.

Results

Model Evaluation and Validation

The app and the model performed better than expected where a random image was uploaded, and it is category was displayed. There were some instances when the model predicts the wrong class which can be due to the image not correctly pre-processed into the right shape or the data set was not large enough. The weights obtained from predictions were matching the expected. Less than 0.5 for the broken almond and bigger than 0.5 for normal. However, in some cases normal almond was classified as broken which can be due to other categories of almond being classified into either normal or broken. Which can be solved by wisely choosing the uploaded image to be either normal or broken almond.

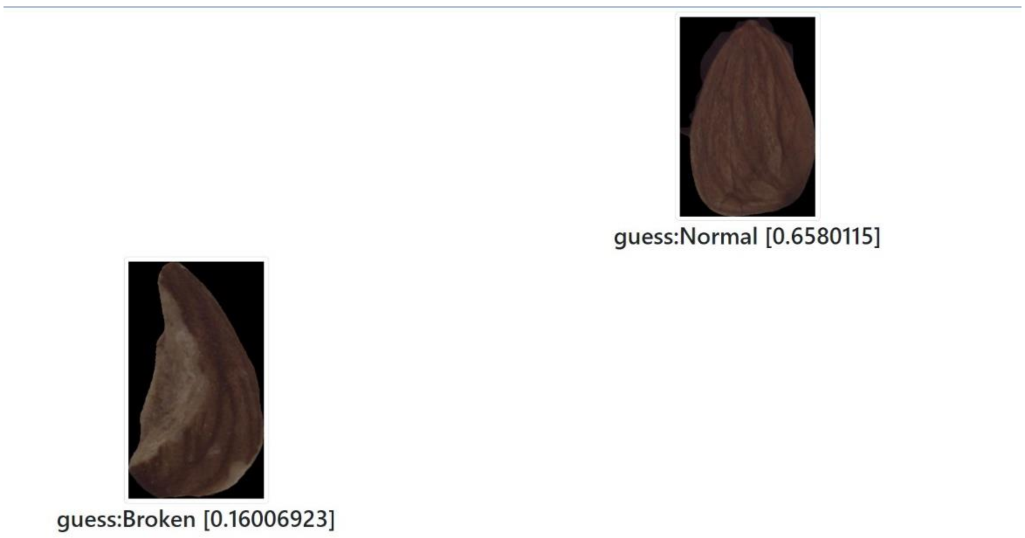


Figure 6 results

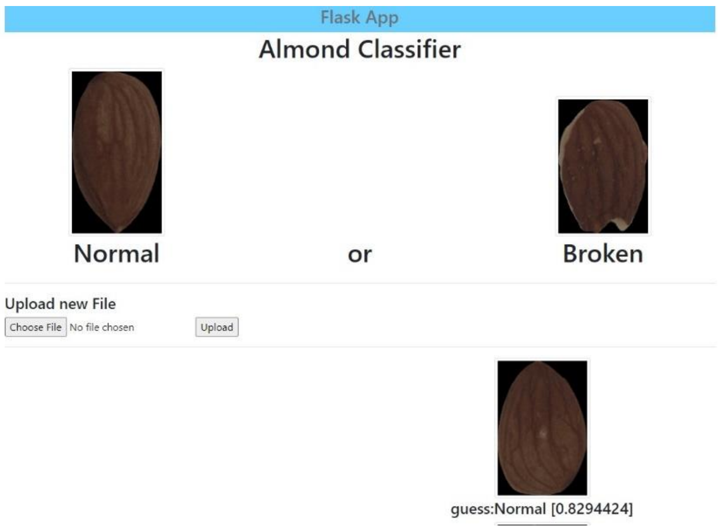


Figure 7 App

As the model was trained for different hyperparameters, the accuracy was recorded for comparison as can be seen in the table below.

Hyperparameters	Accuracy	Val_accuracy
No.images: 10 Epochs : 5 Batch:1 1 CNN	50%	60%
No.images: 20 Epochs : 5 Batch:1 2 CNNs Drop out	70 %	80%
No.images: 20 Epochs : 50 Batch:10 2 CNNs Drop out	95%	90%

From below plots, the model accuracy and loss were illustrated to show how the model performance increased drastically by tunning hypermeters to git the best optimal accuracy.

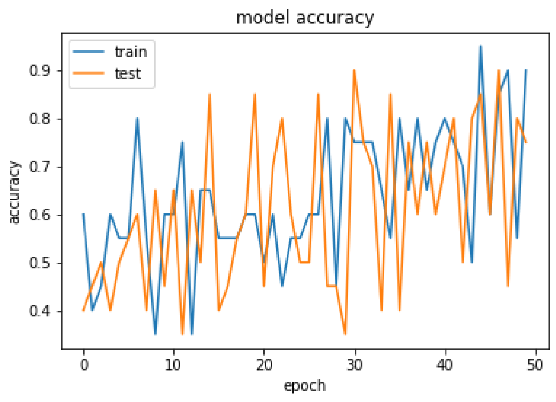


Figure 8 : Accuracy

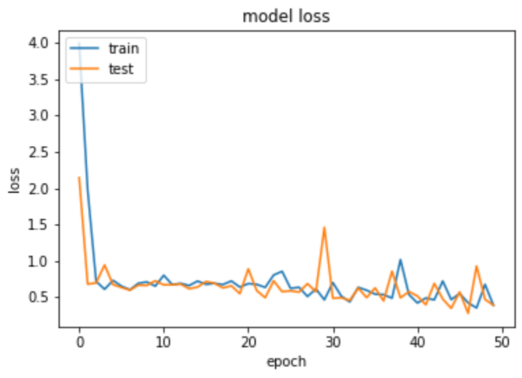


Figure 9 Loss

Justification

It is believed that by adding another convolutional layer, activation layer, dropout layer and increasing the number of epochs has affected the accuracy positively. The model was also able to predict most of the input image as it had similar features to the training dataset. Nevertheless, for some random google image with unique features it was not performing well which can be because of the robustness of the data set.

Conclusion

Improvement

One potential improvement is to use transfer learning and utilize pre trained models like VGG16. Which can show more accurate predictions with less epochs. Another improvement can be by adding extra almond categories such as mould, shrivel and stain classes. Which can add more diversity and robustness to the data set.

References

Brownlee, J., 2022. *Binary Classification Tutorial with the Keras Deep Learning Library*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/binary-classification-tutorial-with-the-keras-deep-learning-library/> [Accessed 20 May 2022].

Blog.keras.io. 2022. *Building powerful image classification models using very little data*. [online] Available at: <https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html> [Accessed 19 May 2022]

Gist. 2022. *Updated to the Keras 2.0 API.*. [online] Available at: <https://gist.github.com/fchollet/0830affa1f7f19fd47b06d4cf89ed44d> [Accessed 20 May 2022].