Building & Training a Neural Network

for Food-Image Classification

**I**

**Abstract**

   This project aims to use deep learning methods to identify images of food (from the food101 dataset) by computer vision. The eventual thrust of the project is to continue the research to develop a competent classifier for food items which could be used in the backend of a health-food-app. The approach was to utilise Tensorflow with Keras to build a model of a convolutional neural network. Three designs were built (based on features ascertained from current research and architectures in the field) and assessed by their validation accuracy. The best model was chosen, and the hyperparameters fine-tuned by means of cross validation. The chosen model was found to be a sequentially built network with 6 convolution layers and 3 Max pooling layers, it was designed to have increased depth. This model was then trained and validated for 15 classes or types of food. The chosen model was then evaluated using the test data. A final training accuracy of 93.3%, a final validation accuracy of 90.9%, and a test accuracy of 66.6% was achieved by our CNN architecture.

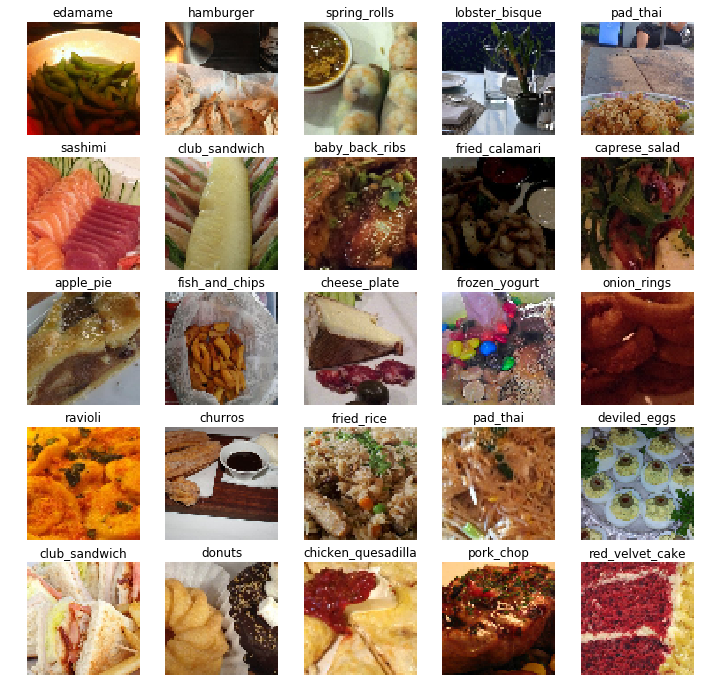
**II**

**Overview**

  Healthy eating is at the heart of a healthy lifestyle. Many App-users already scan barcodes of food items to track the amount of calories they’ve eaten. A Food App which could identify the food item just from an image, would allow for a much easier consumer experience. So, this project focuses on developing techniques to identify types of food items. Further research would allow the quantities of food to also be ascertained and then we could relay the caloric content of those items to the user – just from a picture! This would be the overall scenario were this project to be developed further.

  The dataset we will use is publicly available online, and is called the food101 dataset (TensorFlow. (n.d.). The dataset contains 101 different types or classes of images of food items. Each class contains 250 test images and 750 training images (we will, however, further split the test images into two sets of validation test data). Overall we have 101, 000 images at our disposal. Each time one of our network models is trained or tested on this data, we will do so using a certain number of classes, e.g. if we trained the model for 4 classes we will have used 4 x 750 images. Let’s take a look at some of the images in our dataset in **Figure 1**.

**Figure 1** Some example images with labels from the food101 dataset



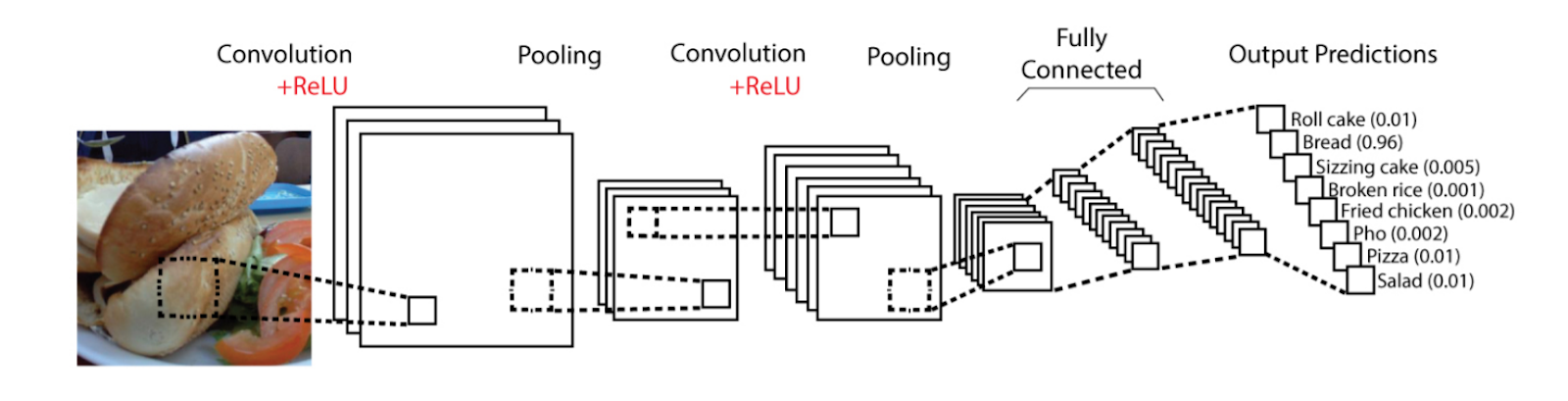
      As we can see from Figure 0. there is a wide variety of images depicting food items of many different categories, from different angles, different distances (zoomed in/zoomed out), different background, different image quality and different plates or surfaces upon which the food is served.

**III**

**Technique**

How deep learning uses Convolutional Neural Networks (CNNs) to identify features and predict classes/types of image:

   CNNs treat each element of its 3x3 filter as a *learn-able* parameter known as the weights of the network (Deeplearning.ai, 2017). The neural network creates its own values for each of the 9 parameters of the filter, based on criteria it has learned will help in discerning between the images fed into the network during training. In this way the network is able to detect *hidden* features of images. The CNN is called so, because it uses convolutional layers in its network - often in addition to other layers: fully connected, pooling, etc. The fully connected layer is one in which all neurons of the network are connected to one another via an input and output - this results in a very large number of parameters (time consuming and memory intensive!). The convolutional layers use the filter to mediate connections between input and output resulting in fewer parameters and faster computation, generally. The fully connected layers (called ‘dense’ layers in Keras) are necessary to learn the functions which classify our data and are normally inserted as the last layers of the CNN, whereas the convolutional layers deal more with feature extraction. Another important aspect of this method is the pooling layer. We can see an example of a typical structure for a CNN in figure 2, where an image of some bread is being fed through a model with a few convolutional and pooling layers before a fully connected layer which feeds through the network’s output to make a prediction of the image’s class.



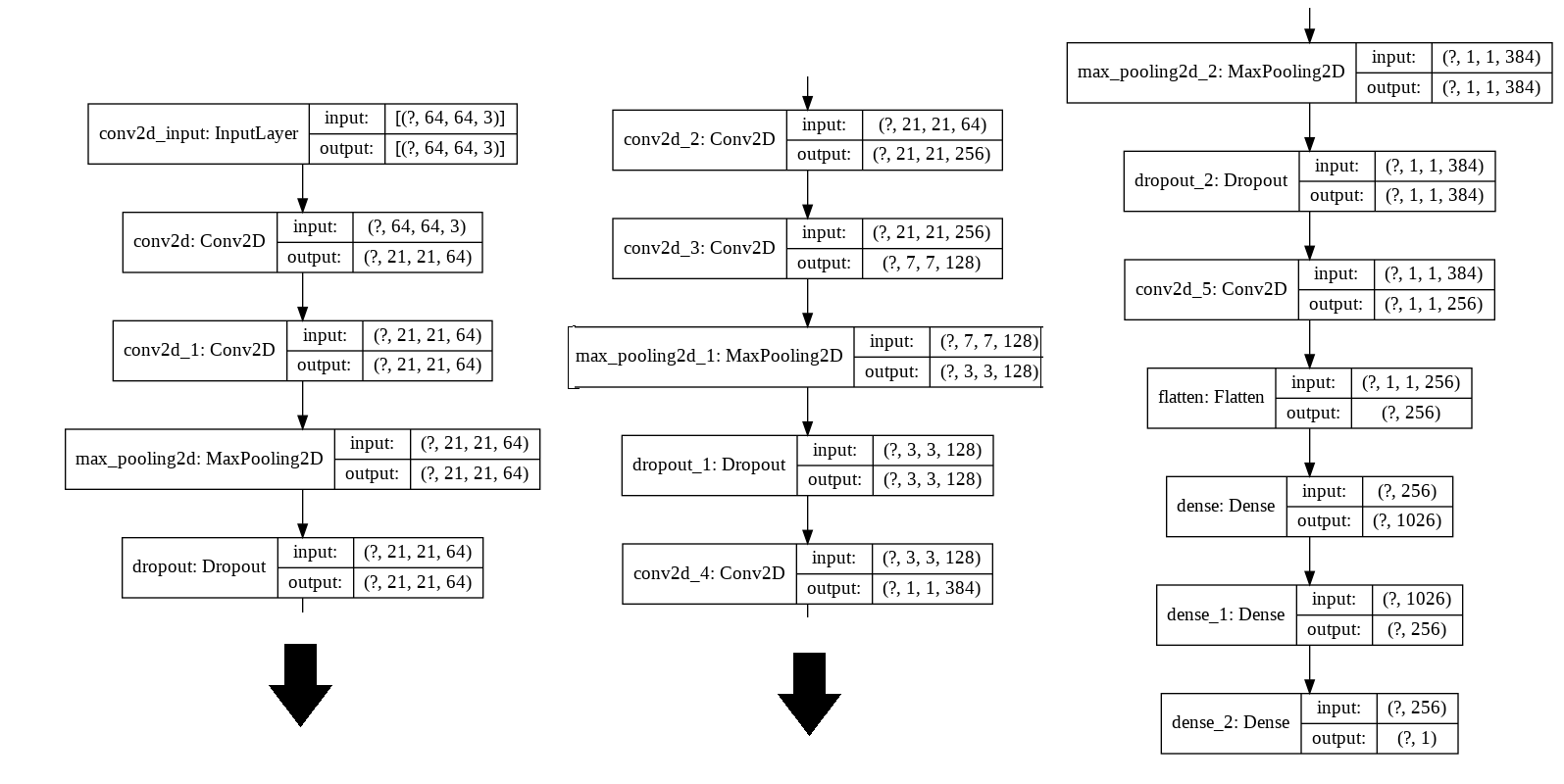
**Figure 2:** A CNN predicting the class of an image depicting a bread roll

**IV**

**Chosen Model**

**No. of parameters: 3,690,195**

Our chosen model contains many more layers than the first design. It was inspired by some of the features discussed in the section on the VGG network. It has 6 convolutional layers, 3 max pooling layers, and 3 dropouts. The final stack of layers are 3 fully connected ones preceded by a flattening of the input. It has a total of 3,690,195 trainable parameters. The increased depth and increased parameters in this network are intended to improve final test accuracy. This model was also built sequentially using Keras. A schematic of the design shows more clearly how it is constructed (see Figure 3). Please note that the network design is one long



**Figure 3:** Diagram of Deeper model #2 Architecture

continuous sequence of layers, the diagram was split into 3 segments to save space on the page.

Finally, training our chosen model on 15 classes of image/type food yielded decent results as shown in the below table, and the final test accuracy was approximately 66%, showing that more work needs to be done to improve our model and reduce overfitting issues.

