**REPORT CA2**

Sidhant Naveria, Abhineet Misra

Institute of Systems Science, National University of Singapore, Singapore 119615

**Abstract**

In today’s world eating healthy food has become very important in order to sustain stress, look good and develop immunity to fight of diseases.

We propose a machine learning model which can classify food as healthy or junk based on its image. This model can we integrated to mobile phones which users can use via app to find the healthy food and avoid junk food

**Introduction**

Every time a person goes out to eat food either in a hawker centre, food courts or buffet in a get together they are not completely sure is the food on their plate is healthy or unhealthy. We are trying to build a model which will help consumer to predict whether the food is healthy or not. This is classification-based problem and we will use a deep learning CNN to tackle this problem. We will extract images of healthy and unhealthy food from google and divide them into training and test set.

Model will be trained on training set and its performance will be checked on the testing set.

**Tools/Techniques**

**Data Extraction**

First problem we face is extraction and collection of data. This is done via python script which crawls the google images and downloads the required data according our problem statement. In total we have collected around 1000 images for both classes (healthy food, junk food).

Next problem is the data images that we have collected varies highly in their sizes for this purpose we have resized each image to our custom need(255 by 255).This image size suits us best as we do not lose much of features from high quality images and this size is computationally inexpensive as well.

Next step is to normalize the data hence we divide each pixel with 255 to normalize it within range of 0 to 1

**Creation of Model**

We will use keras running on tensorflow backhand to create an optimized model

The key idea is to create initial convolution layers to extract important features from data and then pass these features to a set of fully connected layers to do prediction.

We have used varying kernel sizes at every layer along with combination of different pooling and batch normalization

We have tried different activation functions in multiple permutations but Relu gives us the best results.

**Problems Faced:**

**1:** Overfitting: We faced the problem of overfitting here the model was performing well on training data but not so on testing images to counter this problem we used the augmentation of images which randomly augments the images in our training set. Other than that, we used the idea of L2 regularizer to penalize the high values of training weights and the idea of dropouts to let every node participate in the prediction outcome

**2:** Slow learning: We found out the model was taking too long to converge the loss hence we started training the model with a high learning rate as well as low number of batch sizes for the first 40 epochs to quickly reach near the optimal solution

**3:** Not reaching most optimal stage: Towards the end of training the loss was not going down as expected so we used a very small learning rate along with a big batch size for last 60 epochs so that our model can reach the most generalized and optimized state

**Performance of the model:**

In the end the model performs quite well with the minimum accuracy of 88.7 percent on test data. This accuracy increases to 92 percent based on reshuffling of training and test data.

