Challenge – Food Non-Food Classification

By

Lubnaa Abdur Rahman

**Challenge**

# Scope of the Project

Adhering to a healthy and balanced diet can decrease the risk of non-communicable diseases, such as type-2 diabetes, heart disease and cancer. In order to monitor and assess a person’s dietary habits, real-time, intuitive, accurate and cost-efficient dietary assessment approaches are of utmost importance. With the advances in Artificial Intelligence (AI) and machine learning (ML), systems that can automatically output the energy and the nutrients of a consumed meal based on food-related multimedia data (e.g., RGB images, video), have replaced traditional methods, such as 24-hour recall and food frequency questionnaires. The first step of an automatic system is usually the distinction between images that contain food items and images that are irrelevant for this task.

# Data

The dataset consists of 3583 food images from the UNICT-FD889 dataset [1] and 4804 food and 8005 non-food images downloaded from Flickr [2].

# Experiment

You will have to train a CNN in order to perform food/non-food classification. You can select a pretrained CNN or create your own, and also augment the data. Additionally, you will have to output the class activation maps (cam) or gradient class activation maps (grad-cam) of your model, to visualize where the CNN is “looking”.

# References

1. Farinella, G. M., Allegra, D., Stanco, F., & Battiato, S. (2015, September). On the exploitation of one class classification to distinguish food vs non-food images. In International Conference on Image Analysis and Processing (pp. 375-383). Springer, Cham.
2. Farinella, G. M., Allegra, D., & Stanco, F. (2014, September). A benchmark dataset to study the representation of food images. In European Conference on Computer Vision (pp. 584-599). Springer, Cham.

**Work**

*This report shows the important outputs (only) with some brief discussions about the process and the results. The notebook including the full code and all outputs can be accessed* [*here*](https://github.com/lubnaa25/Food_NF_Classification/blob/main/Finale.ipynb)

*The task was completed using Python programming via Jupyter notebook accessed via Kaggle since it offers in-built GPU which was used to reduce image processing and training times. Requirements for the output to be reproduced have been listed in the* [*ReadMe*](https://github.com/lubnaa25/Food_NF_Classification/blob/main/README.md)*.*

*For the CNN model, I have opted to build my own & I have implemented grad-cams to check where the model is looking.*

1. Data Analysis

Once the data was imported, the first step was to check the data distribution to ensure that the data is not biased towards a certain class Training Set

1. Training Set

There was a total of 13113 training images which were split in 1(Food) with 6709 images and 0(Non-Food) 6404 images. As seen from the bar chart, the data distribution is slightly skewed towards Food class, but it is not a huge gap. To ensure that the training is optimized to ignore data skewness, small batches of data have been used for training (64) and during the final model building a uniform distribution is opted for.

Graphical user interface, text, application, email

Description automatically generated

Chart, bar chart

Description automatically generated

1. Test Set

The same was done for the test set and here there was 3279 images with 1678 being for food and the rest being for non food. Even though there is a slight skew towards food class, it is not a considerable one.

Chart

Description automatically generated with medium confidence

Chart, bar chart

Description automatically generated

1. Data Transformation
2. Training Set

Image augmentation was used for preparing the data. ImageDataGenerator is used to transform the images; it takes in the current/original data and applies the transformations as specified, and then returns the transformed image only [1][2].

ImageDataGenerator(

rescale=1./255, #to transform every pixel value from range [0,255] -> [0,1]

shear\_range=0.2, #randomly applying shear transformations - shifting pixels - changing the shape and size of the image accross the axes

rotation\_range=90,

horizontal\_flip=True,#flipping horizontally

fill\_mode='nearest')

Also, the images were resized into shape 224x224x3 [3]

The content of the training set was also visualized along with their corresponding classes as shown below. It is seen that the images (especially objects) are of good quality which will be helpful during the training process.

Graphical user interface

Description automatically generated with medium confidence

1. Test Set

For the test set only rescaling is applied and the shape is resized to 224x224x3 (augmentation is not applied). The visualization of the test set was as follow:

A collage of different foods

Description automatically generated with low confidence

1. Model 1
2. Building

Once the data was ready, a CNN model was built from scratch with the following architecture. A summary and model plot is shown.

Table

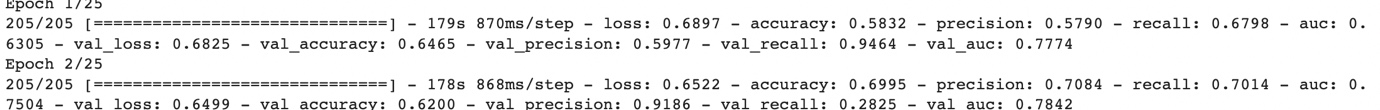
Description automatically generated

Diagram

Description automatically generated

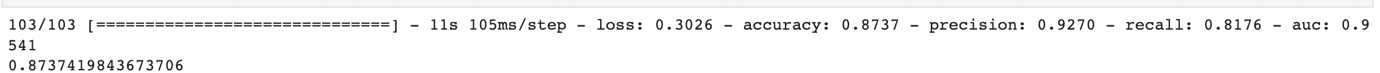
1. Fitting

Model fitting is the process where the model is being trained. Here the training and validation happen simultaneously. The training started with a rather low training and validation accuracy. But the training accuracy reached up to 86.79% while that of the validation reached a maximum of 89.17 %. A total of 25 epoch were run.



1. Evaluation

Then the model was evaluated on the test set giving following results. Even though the model reached an accuracy of 87.3 %(1dp) during the evaluation process, it seems to have a considerable loss and it a much lower loss is desirable. But definitely the model shows promise; for example, it has a high AUC value of 0.9541 which is highly desirable [4]. At this stage the precision and recall are not taken into consideration.



To get a better view of the performance the iterations were plotted. Here the validation accuracy is, on average, higher than the test set (This usually happens when dropouts are being used or because the data has different distribution in each batches of data being taken) For the loss as well, the training loss is more than that of the validation one. Also, quite few peaks are seen for the validation accuracy and the validation loss and this might imply on the stability of the model.

*A more complex model can help with this*

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

1. Tuned Model
2. Model Building

A second model is created and built-upon the first one with an architecture as below. Here a few changes were made; one convolution layer was added, another dropout was added to prevent overfitting and upon building the model the kernel initializer is set to consider a uniform distribution. Also as a means to tune, the optimizer was changed from SGD to Adamax.

Table

Description automatically generated

Diagram

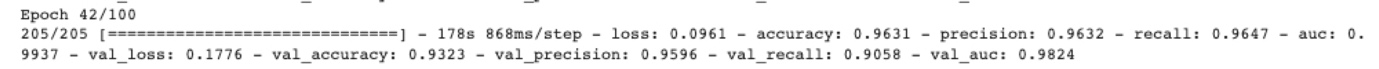
Description automatically generated

1. Fitting

The number of epochs was also increased here to ensure that the model can actually complete training properly. Here an early stopping limitation, as a method for regularization, was set such that if the validation accuracy does not get better over 10 epochs training will be stopped. This is to prevent model overfitting [5]. At this stage a total of 42 iterations were completed. What is seen here is that right from the start the training accuracy and the test accuracy are higher than with the previous model. The **lowest loss** recorded during the validation process is of **0.1673** while the **highest accuracy** recorded was **94.30%**

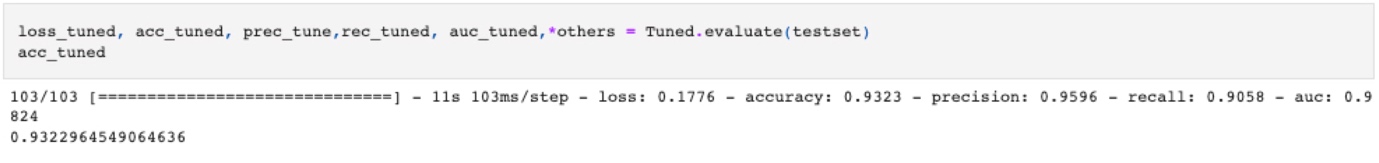
Text

Description automatically generated



1. Evaluation

During the evaluation on the test set, a **loss of 0.1776**, which is comparatively lower than in model 1, and a **good accuracy of 93.23%** was obtained. Additional metrics like auc, recall and precision were computed. But at this stage it is noted that the **AUC**, which suggest the discriminative ability of the model, has value of **0.9824** is quite admirable and is quite desirable [6]



1. Plots

On the accuracy plot, you can see that the model’s accuracy generalizes after epoch 32 and there is no improvement. Here the training accuracy and the test accuracy seem to be quite close to each other indicating that the model is not being overfitted/underfitted. While there are certain peaks for the validation(testset) accuracy, this is expected as at different epochs different types of data is being presented to the model.

For the loss, as it may be seen, the values are rather low and after epoch 32 as well, the loss does not improve. So, to prevent overfitting the model, training is stopped automatically.

Chart, line chart

Description automatically generated

Chart

Description automatically generated

1. Models‘ comparison

As previously seen, the loss has been decrease by a value of 0.125 while the accuracy increased by 5.86%. Based on these figures, it can be said that the tuned model proves to be better at making the classifications and is therefore chosen as the final model. Definitely, even if the tuned model’s accuracy (evaluation stage) is quite high, there is always room for improvement.

Graphical user interface, application

Description automatically generated

*Plots*

When both models’ performance across epochs were plotted on the same graph, it can be clearly seen that both in terms of accuracy and loss the “Tuned” model did a better classification right from the start of training and validation.

Chart

Description automatically generated

Chart, histogram

Description automatically generated

1. Prediction

*The next stage is to carry out predictions using the tuned model. Here, predictions are carried on the test set and visualized.*

Once the predictions are done, the mean of their distribution is visualized to see how the model has classified the cases in the test set. It is noted that, as seen in section 1, there were 1678 Food cases and 1601 non food cases. First, it was seen that the mean of the predictions was of 0.48 and the distribution was slightly skewed towards non-food cases.

Chart, bar chart, histogram

Description automatically generated

Next the actual images in the test set were visualized[[1]](#footnote-1) with their corresponding probabilities of predicted class and also actual class. It is seen here that out of the three, only the 1st and 3rd predictions were correct and these were with high probabilities. While the middle picture was incorrectly classified as non-food even though it was actually a food case.

A collage of food

Description automatically generated with low confidence

Additionally, a confusion matrix was built for the model to show what the actual and predicted cases were. From this the precision and recall for the classification of food items can be deducted. At first glance it can be seen that the model does a slightly better job at classifying non-food items compared to food ones. Now let’s look at the TPR, TNG, FPR, FNR.

Graphical user interface

Description automatically generated

Note: +ve is considered as class 1, -ve is considered as class 0. To ensure that the model is doing a good job in classification TPR and TNR should be high while FPR and FNR should be low. So TPR high means the model does a good job at classifying Food, TNR high means the model does a good job at classifying Non-food. And low FPR mean that the model classifies a low of non-food wrongly while low FNR entails that the model is classifying a low amount of food wrongly.

So here, looking at the numbers, you can see that these rates are quite high and it is confirmed that indeed the model is slightly better at classifying non-food cases. It is also seen that the model classifies non-food items less incorrectly as opposed to food items. But these numbers are rather low and can be acceptable [7]. Other important metric which is looked at is the precision which demonstrates the model’s capability of identifying a food case and here it has reached a score of 95.95%, and also the recall, which is the measure for the food cases correctly classified by the model among all the food cases, of 90.58%.

Total: 1678 Food (+VE) cases and 1601 (-VE) non-food cases

**TPR/Recall =1520/1678 = 90.58 %**

**TNR = 1537/1601 = 96.00 %**

**FPR=64/1601=4.00%**

**FNR=158/1678=9.42%**

**Precision=1520/1584=95.95%**

1. Grad-CAMs

To better understand why wrong predictions, or even right ones, are made, it is important to understand what the model is looking at to make its predictions. For that, gradient class activation mappings (Grad-CAMs) have been used to demonstrate visual explanations for the model’s predictions [8]. The code for the gradcam can be seen in the notebook.

Note: only the resulting gradcam for the last convolution layer is shown.

Here the predicted class and the actual class are shown along with the original image, a heatmap built and a superimposed image showing the gradients. You can see that the food part is actually being highlighted. Even though it is not fully exactly on the middle where the food (like border of plat is also highlighted) but the model is looking at the correct place. You can see the back of the picture is being ignored (black color in middle image, blue color in last image)

A picture containing table

Description automatically generated

This is an example of a non-food case which was correctly classified. The parts not looked at are in black in second image, and in blue in the third image. Parts being looked at where its detecting the non-food is being highlighted in the reds, yellow, green parts.

A picture containing application

Description automatically generated

Another image of food correctly classified, here the probability of the predicted class is also shown for reference. The food parts seems to be clearly highlighted too especially the ones in the foreground.

Graphical user interface

Description automatically generated

In this image, the heatmap produces gradients mostly for the upper half of the image while the left bottom end is ignored. As such, the model has correctly identified this case as a non-food case with a very high probability.

A picture containing application

Description automatically generated

Here is another case where food was correctly classified. You can see that the model is looking right into the middle (in red) and identifying the food.

A picture containing application

Description automatically generated

The next image is a case incorrectly classified. In the test set this picture was labelled as food however the model says it is non-food. Now we see that the model is looking at the bottom corner of the picture only. And indeed, there is no food in the bottom end only while the rest of the image is ignored. Only a plate and bowl. So, in this way the model might be correct. Also, if the image in itself is looked at, food can be barely distinguished.

Graphical user interface

Description automatically generated

Next, is a case where food was correctly classified. In this instance the bowl of food was only half full and the model has successfully identified where there is food on the image and looked only at that part to make the prediction.

A picture containing Excel

Description automatically generated

Note: More images for the gradcam examples can be found in the [notebook](https://github.com/lubnaa25/Food_NF_Classification/blob/main/Finale.ipynb)

1. Outcomes

As the challenge was successfully completed as per requirements (data augmentation, model building & training, grad-cam), the following were the major findings & outcomes

1. A preliminary data exploration was done to check the data distribution which for both training and test sets were slightly skewed towards the food class. But the differences were not massive. To prevent class biasedness especially during training, small batches of data were used & uniform distribution was taken into consideration (Tuned model).
2. The images were augmented by applying transformations using Image Data Generator.
3. A first model was built which showed quite some promises, but a lower loss was desirable. Therefore a “tuned” model which was a bit more complex in terms of the architecture was built to prevent underfitting and a dropout layer was also added to prevent overfitting and the optimizer was changed from SGD to Adamax.
4. The tuned model gave superior performance with a high accuracy of 93.23 % And a loss of **0.1776.** From the confusion matrix the model was found to be 95.95% precise with a recall of 90.58%. The model was slightly better at identifying non-food cases as opposed to food cases.
5. As some incorrect predictions were there, grad-cams were built to give a better view at what the model was looking. In most instances shown, the model identifies food and non-food correctly.
6. However, the problem seems to arise in pictures where utensils are involved. It was identified that in the test set one of the images showed mostly empty bowls and therefore the model returned the prediction as non-food, which is not entirely wrong, even though the test set labelled it as food. So, in the future the dataset could be reviewed.
7. While the model did show good performance, there is always room for improvement and more exploration needs to take place on how to make it better for food classification.

References:

[1] R. Adrian, “Keras ImageDataGenerator and Data Augmentation,” *PyImageSearch*, Jul. 08, 2019. <https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/> (accessed Apr. 25, 2022).

[2] Keras, “Building powerful image classification models using very little data.” <https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html> (accessed Apr. 25, 2022).

[3] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, and Y. Ma, “DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment,” in *Inclusive Smart Cities and  Digital Health*, Cham, 2016, pp. 37–48. doi: [10.1007/978-3-319-39601-9\_4](https://doi.org/10.1007/978-3-319-39601-9_4).

[4] J. N. Mandrekar, “Receiver Operating Characteristic Curve in Diagnostic Test Assessment,” *Journal of Thoracic Oncology*, vol. 5, no. 9, pp. 1315–1316, Sep. 2010, doi: [10.1097/JTO.0b013e3181ec173d](https://doi.org/10.1097/JTO.0b013e3181ec173d).

[5] H. Liang *et al.*, “DARTS+: Improved Differentiable Architecture Search with Early Stopping,” *arXiv:1909.06035 [cs]*, Oct. 2020, Accessed: Apr. 25, 2022. [Online]. Available: <http://arxiv.org/abs/1909.06035>

[6] “Diabetic Retinopathy: Present and Past - ScienceDirect.” <https://www.sciencedirect.com/science/article/pii/S1877050918308068> (accessed Apr. 25, 2022).

[7] A. S. Jadhav, “A novel weighted TPR-TNR measure to assess performance of the classifiers,” *Expert Systems with Applications*, vol. 152, p. 113391, Aug. 2020, doi: [10.1016/j.eswa.2020.113391](https://doi.org/10.1016/j.eswa.2020.113391).

[8] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization,” *Int J Comput Vis*, vol. 128, no. 2, pp. 336–359, Feb. 2020, doi: [10.1007/s11263-019-01228-7](https://doi.org/10.1007/s11263-019-01228-7).

1. Visualizations were done randomly [↑](#footnote-ref-1)