Assignment 4

av

Lars-Erik Bakkland Moi and Oskar Markussen

i

IKT450
Deep Neural Networks

Universitetet i Agder

Grimstad, August 2023

Contents

1 Task

In this task you are suppose to implement a convolutional neural network using a high level library, e.g. PyTorch. The classification should be about food.

- Download the food 11 dataset https://mmspg.epfl.ch/food-image-datasets or https://www.kaggle.com/vermaavi/
- Predict the 11 classes: Bread, Dairy product, Dessert, Egg, Fried food, Meat, Noodles/Pasta, Rice, Seafood, Soup, and Vegetable/Fruit
- Try some standard networks convolutional networks before more complex ones.

2 Convolutional Neural Network (CNN)

The objective of this project was to build and train a Convolutional Neural Network (CNN) for classifying images into 11 different food categories. The dataset used for this task consisted of images categorized as Bread, Dairy Products, Dessert, Eggs, Fast Food, Meat, Pasta/Noodles, Rice, Seafoods, Soup, and Vegetables/Fruits. This dataset was divided into training and validation sets for model training and evaluation.

During preprocessing, several steps were taken to prepare the data. Firstly, the label for each image was extracted from the filename using a function 'get_label'. Secondly, the images were read from their file paths, decoded into JPEG format, resized to a standard size of 150x150 pixels, and normalized to have pixel values between 0 and 1 through the 'process_path' function. To augment the training data and introduce diversity, data augmentation techniques such as random horizontal flipping and random rotation of 10% of the image were applied.

The CNN model designed for this task included an input layer expecting images of size 150x150x3, followed by four convolutional layers with increasing numbers of filters (32, 64, 128, and 256). Each convolutional layer was followed by Batch Normalization and Max Pooling layers. A Global Average Pooling layer was used to reduce spatial dimensions before connecting to a dense layer with 512 units, L2 regularization, and a dropout layer for regularization. The final output layer had 11 units corresponding to the 11 classes and used softmax activation.

For training, the model was compiled using the Adam optimizer and sparse categorical cross-entropy loss, with accuracy as the evaluation metric. Early stopping with a patience of 100 epochs was employed to prevent overfitting. The model was trained for up to 100 epochs.

The model's performance was evaluated using a confusion matrix constructed from predictions on the validation set. This matrix was visualized using a heatmap. Additionally, plots of the model's accuracy and loss over the training epochs were generated to assess performance and convergence.

The results indicated that the trained model achieved an accuracy of 72%, as visualized in both the confusion matrix and the accuracy plot.

```
Epoch 93/100
309/309 [==============================] - 8s 24ms/step - loss: 0.4822 - accuracy: 0.8954 - val_loss: 1.6956 - val_accuracy: 0.6443
Epoch 94/100
309/309 [==============================] - 8s 26ms/step - loss: 0.4704 - accuracy: 0.8991 - val_loss: 1.4028 - val_accuracy: 0.6930
Epoch 95/100
309/309 [==============================] - 8s 26ms/step - loss: 0.4472 - accuracy: 0.9073 - val_loss: 1.2704 - val_accuracy: 0.7120
Epoch 96/100
309/309 [==============================] - 8s 26ms/step - loss: 0.4398 - accuracy: 0.9073 - val_loss: 1.6082 - val_accuracy: 0.6493
Epoch 97/100
309/309 [==============================] - 8s 25ms/step - loss: 0.4572 - accuracy: 0.9011 - val_loss: 1.4781 - val_accuracy: 0.6924
Epoch 98/100
309/309 [==============================] - 8s 25ms/step - loss: 0.4552 - accuracy: 0.9031 - val_loss: 1.2747 - val_accuracy: 0.7274
Epoch 99/100
309/309 [==============================] - 8s 26ms/step - loss: 0.4485 - accuracy: 0.9081 - val_loss: 1.5017 - val_accuracy: 0.6927
Epoch 100/100
309/309 [==============================] - 8s 25ms/step - loss: 0.4561 - accuracy: 0.9035 - val_loss: 1.2823 - val_accuracy: 0.7242
108/108 [==============================] - 2s 18ms/step
```

Figure 1: Accuracy Testing

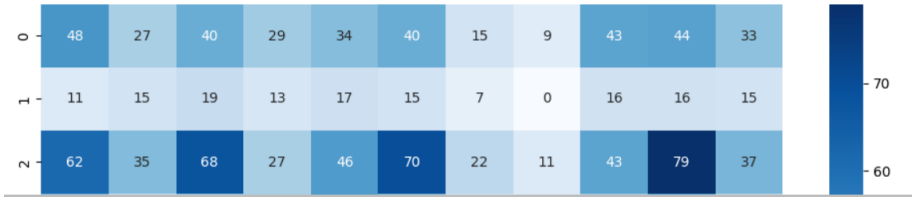

Figure 2: p1 confusion matrix

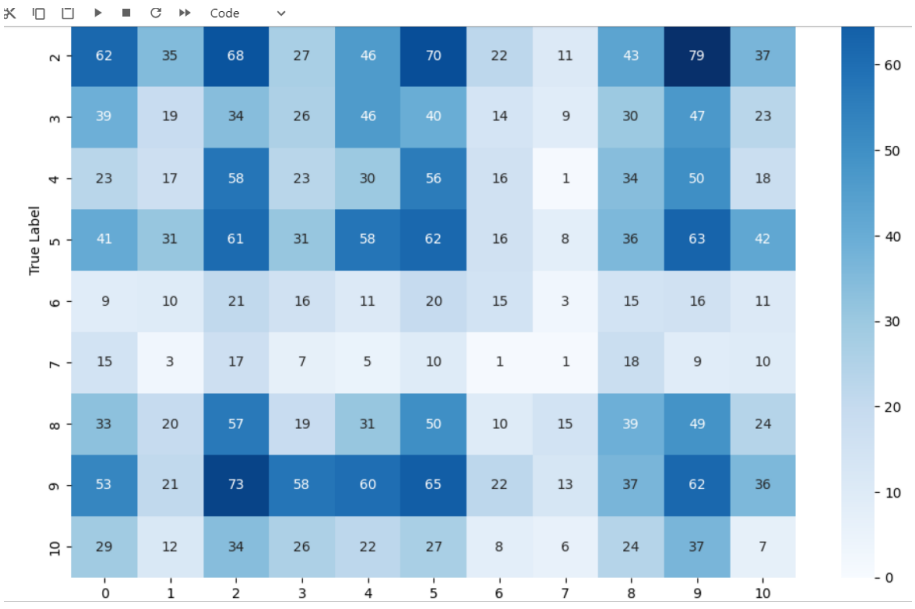

Figure 3: p2 confusion matrix

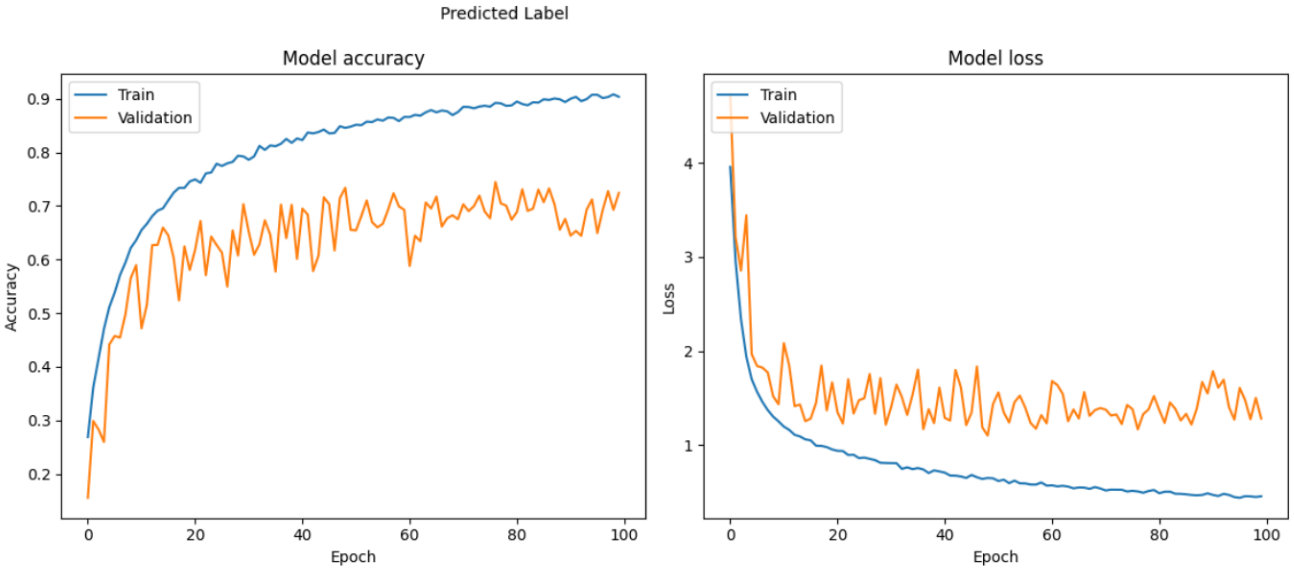

Figure 4: Accuracy and Loss graph