Assignment 2

**Transfer Learning**

Name: Cedric Oliveira da Silva Costa  
Email: [cedric.oliveiradasilvacosta@connect.qut.edu.au](mailto:cedric.oliveiradasilvacosta@connect.qut.edu.au)  
Student Nr.: n11369574

Name: Abhishek Sapkal  
Email: [abhishekdilipsapkal@connect.qut.edu.au](mailto:abhishekdilipsapkal@connect.qut.edu.au)  
Student Nr.: n10831908

**IFN680   
Artificial Intelligence and Machine Learning**  
  
Dr Frederic Maire

Queensland University of Technology   
30.10.2022

# Introduction**:**

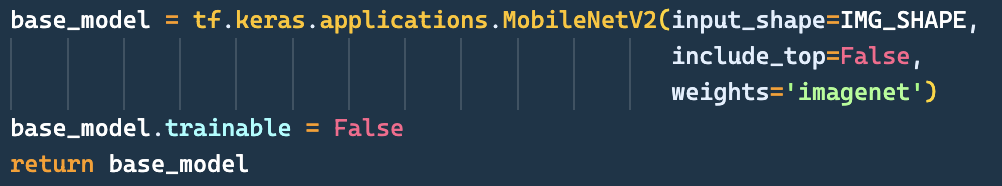
In this assignment, we are performing and building an image classification (flower classifier) using the application of machine learning. Pre-trained models such as convolutional neural networks containing learned features such as corners, edges, shape, colour, etc. are implemented.

# Task 1 – Download dataset:

The dataset was downloaded and saved locally with the folder-name: ‘small flower dataset’.

# Task 2 – Download pre-trained MobileNetV2 network:

To use the pre-trained MobileNetV2 network we have to use the **MobileNetV2**-function provided by **tf.keras.applications**. As the **input\_shape** we declared a constant **IMG\_SHAPE** with the tuple value of (224,224,3) as this is one of the sizes that the pre-trained model supports. In addition to that we set the **include\_top** parameter to **False** to get a copy of the model without its classification layers (output layers). As the weights parameter we set **imagenet**.

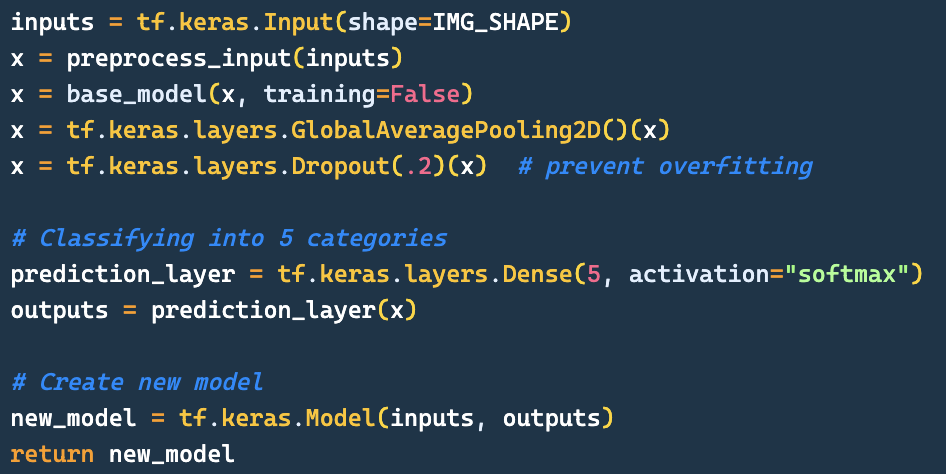


*Figure 1: Download MobileNetV2 model*

Also, as in figure 1 can be seen, we had to set the **trainable** attribute to **False** to freeze the model for our further purposes.

# Task 3 – Replace / Add last layer of the network with a Dense-layer:

As described in task 2 we downloaded the model without its classification layer and therefore we do not have to replace its last layers. Instead, we must add our own classification layers which in our case are the GlobalAveragePooling2D-, Dropout- and Dense-layer (with 5 classes).



*Figure 2: Customize downloaded MobileNetV2 model*

We needed to add the **GlobalAveragePooling2D**-layer due to how the model works with the image data and the **Dropout**-layer was recommended to prevent overfitting. As the task required, we added a **Dense**-layer which supports our 5 classes: daisy, dandelion, roses, sunflowers and tulips.

The added input layer that can be also seen in figure 2 was added to help with mapping pixel values to [-1, 1] as this is the format that the MobileNetV2 model works with.

# Task 4 - Data preparation for non-accelerated version:

To prepare our datasets we used the **image\_dataset\_from\_directory** provided by **tf.keras.utils**. Due to its simplicity, we were able to provide the function with just a few parameters to help us with common data preparation steps needed, like shuffling, batching, labelling and splitting the data. An example for can be seen in the figure below:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

*Figure 3: Example for using the image\_dataset\_from\_*directory-function

In the preperation process a total of 1000 images split into 5 categories were provided for our task which we had to divide into train and test datasets. We decided to perform an 80-20 split, i.e., 800 images are used for training and the remaining 200 images are used for validations. With a batch size of 32 a total of 32 batches were created out of the 1000 images.

A picture containing graphical user interface

Description automatically generated

*Figure 4: Output of our data preparation process*

These 32 batches were split into 25 for training purposes, 6 for validation purposes and 1 for test purposes as shown in figure 4.

# Task 5 - Compile the model:

Once the preparation of training, validation and test dataset is done, the model is then compiled with SGD optimizer. With the parameters **learning\_rate=0.01**, **momentum=0.0**, **nesterov=False**. The accuracy with the compiled model and the test dataset is the following:



*Figure 5: Accuracy value with SGD optimizer*

Since the accuracy is low, the prediction is also very low. In the below image, one can see the out of 24 test images, only 4 have been predicted correctly.

Diagram

Description automatically generated

*Figure 6: Model predictions for non-accelerated version*

# Task 6 – Plotting the graphs:

# **Chart, line chart Description automatically generatedChart, line chart Description automatically generated**

*Figure 7: Training and validation loss v/s Time(epochs)*

The above two graphs provide the data for the training and validation errors vs time as well as the training and validation accuracies. Starting with the left graph, the training accuracy is represented by blue graph, and we could see that after **epoch 7** contains the maximum value of **0.218**. However, the validation accuracy measured on validation dataset and is represented by red graph is very low and which states that the compiled model was unable to classify the images. Moreover, one the right graph, the validation loss is much higher. In both cases **epoch 5** seems to be the best performing due to the increasing variance for every additional epoch.

# Task 7 – Experimenting with three new learning rates:

In this task we should play around with three new learning rates. i.e., **0.00005**, **0.00001** and **0.1**. The model is then trained with this new value and the one which provides the highest accuracy is picked.



*Figure 8: Learning rate with highest accuracy*

The bar plot shows the accuracy/loss value for each learning rates values. Figure 9 shows that the learning rate of **0.1** yields the highest accuracy.

Chart, bar chart

Description automatically generated

*Figure 9: Plotting the accuracy and loss value for different learning rates*

# Task 8 - Experimenting with three new momentum rates with best learning rate with best accuracy:

After the best learning rate was calculated in the previous task, the same process must be repeated with three additional momentum rates. The accuracies resulting due to the different momentum rates are then compared with each other to find the highest value. By using these techniques, we have managed to increase the accuracy level from **0.21** to **0.25** with a momentum rate of **0.001** with can be seen in the following figure:

Chart, bar chart

Description automatically generated

*Figure 10: Plotting the accuracy and loss value for different momentum rates*

In the following visualization the prediction of the model with best found learning-rate of **0.1** and best found momentum-rate of **0.001** is shown:

Timeline

Description automatically generated with low confidenceDiagram

Description automatically generatedTimeline

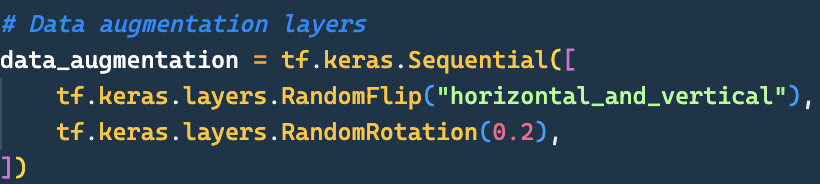
Description automatically generatedDiagram

Description automatically generated

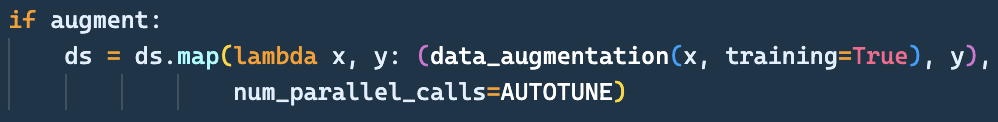
*Figure 11: Model predictions from top left to right for 0.0, 0.001, 0.01 and 0.1 respectively*

# Task 9 – Prepare the datasets with accelerated version:

In our experiment, the model has under-performed i.e., the accuracy value is very much on the lower side. In this step, the data is augmented. The data augmentation is process in which the existing images are changed in minor scale. These changes generally include transformation such as rotation, flipping, rescaling, and resizing.



*Figure 12: Data augmentation function used in the project*



*Figure 13: Applying data augmentation to keras datasets*

As in figures 12 and 13 shown, operations like flipping an image horizontally and vertically and also rotating were applied to the training dataset.

# Task 10 – Experimenting for the accelerated version with new momentum rates:

The same process of compiling the existing model with three different learning rates and with new augmented dataset is done. The accuracy values are compared and the learning rate with highest accuracy is picked, and model is compiled again with three new momentum values.

Chart, bar chart

Description automatically generated

*Figure 14: Plotting the accuracy and loss value for different learning rates (acc. version)*

The using the best learning rate and momentum rate, the highest accuracy is executed. However, the accuracy is again low, and it explains the less prediction as in the figure 14.

Chart, bar chart

Description automatically generated

*Figure 15: Plotting the accuracy and loss value for different momentum rates (acc. version)*

Chart

Description automatically generated with medium confidenceDiagram

Description automatically generated with medium confidenceDiagram

Description automatically generatedA picture containing diagram

Description automatically generated

*Figure 16: Model predictions from top left to right for 0.0, 0.001, 0.01 and 0.1 respectively*