Pattern recognition and machine learning competition task 2 report for team #2

## Testing three different CNN models

Three different CNN models were tested to find the best one for which we would continue experiments to achieve the highest score possible. MobileNet, MobilenetV2 and InceptionV3 were trained with the training data with a 3 epoch setting to get a general idea of how well they compete with each other. It should be noted that the experiment was made by feeding the features extracted in task 1 into the models – and not the images themselves. The 3 epoch test results are presented in the table below.

|  |  |
| --- | --- |
| Model name | Score with just 3 epoch |
| MobileNet | 0.23 |
| MobileNetV2 | 0.16 |
| InceptionV3 | 0.30 |

Table1: 3 Epoch test results

Based on the 3 epoch test results, we decided to put our future investigation into Inceptionv3 model. For the proper classification purpose, we trained the network with the images data which resulted in a good prediction accuracy.

## Essential code parts:

We created a (simple) solution where we fed the whole training images into the InceptionV3 model. and added a 100 node dense layer, and a 17 node output layer.

|  |
| --- |
|  |
|  | base\_model = tensorflow.keras.applications.inception\_v3.InceptionV3(include\_top=False, weights='imagenet', input\_tensor=None, input\_shape=(224,224,3), pooling=None, classes=17)  #Grab the input of the base model  in\_tensor = base\_model.inputs[0]  # Grab the output of base model |
|  | out\_tensor = base\_model.outputs[0] |
|  | out\_tensor =tensorflow.keras.layers.Flatten()(out\_tensor) |
|  | out\_tensor =tensorflow.keras.layers.Dense(100, activation='relu')(out\_tensor) |
|  | out\_tensor =tensorflow.keras.layers.Dense(17,activation='softmax')(out\_tensor) |
|  | model = tensorflow.keras.models.Model(inputs = [in\_tensor],outputs = [out\_tensor]) |
|  | model.summary() |
|  | batch\_size = 50  epochs = 15 |

As can be seen from the code above, we increased our epochs to 15 and fed in (224, 224, 3) dimensional image data to our model.

## Submission to Kaggle platform

Running the code resulted in a local testing score of 86% and a Kaggle score or 83%. Similar to our results in task 1, a ~3% drop in the score can be seen.