LAB 6 REPORT

A PREPRINT

Daniele cecca
Artificial Intelligence for Science and Technology
Milano Bicocca University
Supervised Learning

May 4, 2024

1 Introduction

In this study, we will apply Transfer Learning to various Neural Network architectures, specifically MobileNetV2 and GoogleNet. Both are pre-trained neural networks. For the first model, we will conduct two fine-tunings: the first solely on the added layers, and the second on all the network's layers. To evaluate our networks, we will utilize PLCC (Pearson Linear Correlation Coefficient) and SROCC (Spearman Rank).Ultimately, we will perform a comparison among the models.

2 Transfer Learning

In transfer learning, the network is pre-trained to perform a related secondary task for which data are more plentiful. The resulting model is then adapted to the original task. This is typically done by removing the last layer and adding one or more layers that produce a suitable output. The main model may be fixed, and the new layers trained for the original task, or we may fine-tune the entire model.

3 MobileNetV2

MobileNetV2 is a new neural network architecture that is specifically tailored for mobile and resource constrained environments. The network pushes the state of the art for mobile tailored computer vision models, by significantly decreasing the number of operations and memory needed while retaining the same accuracy. The main innovation is a novel layer module: the inverted residual with linear bottleneck. This module takes as an input a low-dimensional compressed representation which is first expanded to high dimension and filtered with a lightweight depthwise convolution. Features are subsequently projected back to a low-dimensional representation with a linear convolution. This convolutional module is particularly suitable for mobile designs, because it allows to significantly reduce the memory footprint needed during inference by never fully materializing large intermediate tensors.

3.1 Transfer Learning (Last Layers)

In order to apply transfer learning and consequently train our model, we have to:

- Remove the last layer
- Add our new layer
- Decide which layers to freeze and which not
- Train the new model

Thus, we remove the layer denoted by the name "classifier" and we add our layer which will be composed in the following way:

```
(classifier): Sequential(
  (0): Linear(in_features=1280, out_features=512, bias=True)
  (1): GELU(approximate='none')
  (2): Linear(in_features=512, out_features=32, bias=True)
  (3): GELU(approximate='none')
  (4): Linear(in_features=32, out_features=1, bias=True)
)
```

After that, we freeze pre-trained layers and unfreeze the added layer. And at the end, we start the training on our dataset.

For this training, we have used the L1Loss regression as the loss function and ADAM as the optimizer. Also, we have set the number of epochs equal to 2.

3.1.1 Test phase

After the training phase, we have tested this model on the test set and obtained the following results:

Table 1: Performance Metrics

Metric	Value
Test Loss	0.076877
PLCC	0.651
SROCC	0.588

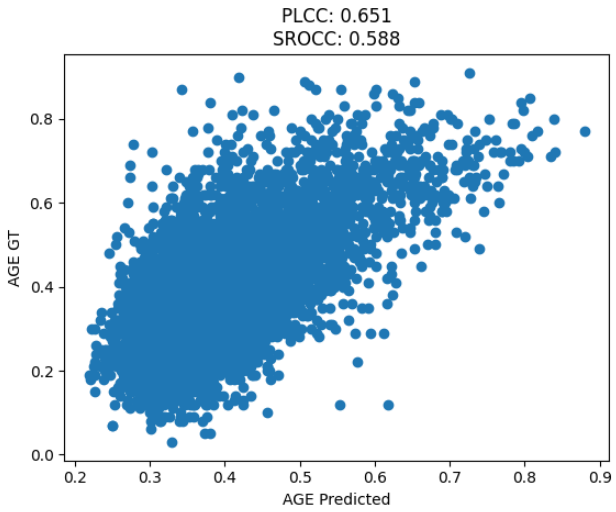

Figure 1: true labels-predictions

3.2 Transfer Learning (All layers)

We repeated the same experiment described before, but in this case, we train all the layers. In other words, we unfreeze all the layers. As a result, we obtained:

Table 2: Performance Metrics

Metric	Value
Test Loss	0.060719
PLCC	0.791
SROCC	0.764

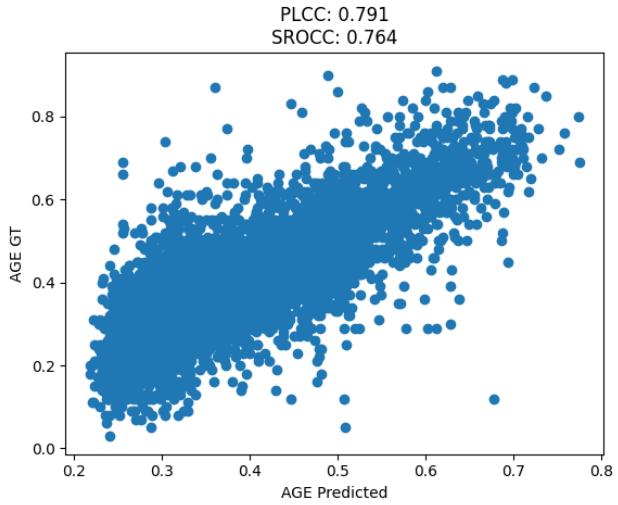

Figure 2: true labels-predictions

4 GoogleNet

GoogLeNet is a type of convolutional neural network based on the Inception architecture. It utilises Inception modules, which allow the network to choose between multiple convolutional filter sizes in each block. An Inception network stacks these modules on top of each other, with occasional max-pooling layers with stride 2 to halve the resolution of the grid.

4.1 Transfer Learning (Last Layers)

We re-proposed the experiment done with MobileNetV2 with the GoogleLeNet network. As before, we have removed the last layer, which is called "fc" in this case, and we added the same layer as before:

```
(fc): Sequential(
  (0): Linear(in_features=1280, out_features=512, bias=True)
  (1): GELU(approximate='none')
  (2): Linear(in_features=512, out_features=32, bias=True)
  (3): GELU(approximate='none')
  (4): Linear(in_features=32, out_features=1, bias=True)
)
```

and we performed the training.

4.1.1 Test phase

After the training phase, we have tested our model and obtained the following results:

Table 3: Performance Metrics

Metric	Value
Test Loss	0.078940
PLCC	0.625
SROCC	0.566

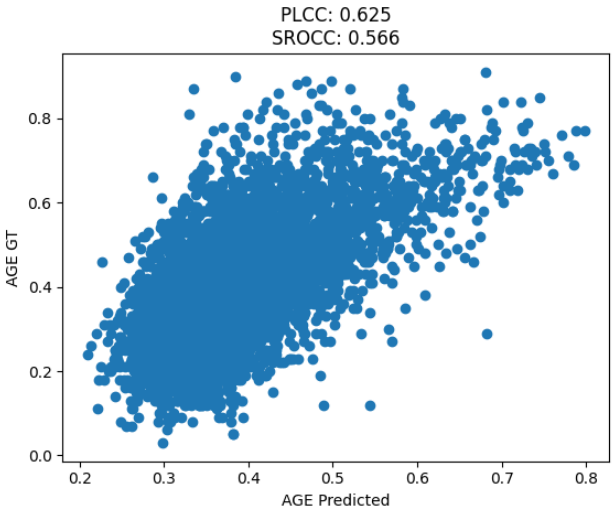

Figure 3: true labels-predictions

5 Conclusion

In conclusion, this study applied Transfer Learning to two different pre-trained neural network architectures, MobileNetV2 and GoogleNet, for a specific task. Two scenarios were explored for the first network: fine-tuning only the last added layers and fine-tuning all layers.

For MobileNetV2, when only the last layers were fine-tuned, the PLCC was 0.651 and the SROCC was 0.588. Fine-tuning all layers resulted in improved performance with a PLCC of 0.791 and a SROCC of 0.764. This suggests that allowing all layers to be trainable leads to better model performance compared to only fine-tuning the last layers.

Similarly, for GoogleNet, fine-tuning only the last layers yielded a PLCC of 0.625 and a SROCC of 0.566. Despite utilizing the same transfer learning approach as MobileNetV2, GoogleNet's performance was slightly lower in terms of PLCC and SROCC.

Overall, the results indicate that fine-tuning all layers generally leads to better performance compared to fine-tuning only the last layers, as observed in both MobileNetV2.

References

[1] Transfer Learning *Understanding Deep Learning*. Page 152-153 Available at: `https://github.com/udlbook/udlbook/releases/download/v2.06/UnderstandingDeepLearning_05_01_24_C.pdf`. Accessed 4 May. 2024.

[2] Introduction MobileNetV2 *MobileNetV2 paper*. Available at: `https://arxiv.org/pdf/1801.04381`. Accessed 4 May. 2024.

[3] Introduction GoogLeNet *GoogLeNet paper*. Available at: `https://paperswithcode.com/method/googlenet`. Accessed 4 May. 2024.