Age classifier with Transfer Learning - Assignment 6

Mirko Morello
920601
m.morello11@campus.unimib.it

Andrea Borghesi
916202
a.borghesi1@campus.unimib.it

January 3, 2025

1 Introduction

In this assignment, our objective was to exploit the weights of CNN architectures trained on a dataset and transfer that knowledge to accomplish a related task.

2 Technologies used

2.1 Convolutional Neural Networks

Convolutional Neural Networks are a type of Neural Networks that do not need tailored feature engineering, but rather, they automatically learn them. The layers specialized in this automatic feature extraction are called the Convolutional layers. They are biologically inspired by the connectivity of the visual cortex in animals, where individual cortical neurons are responsible for a restricted portion of the visual field, known as the *receptive field*. This type of connectivity drastically reduces the connections between neurons compared to the dense layers of a classical fully connected Neural Network. The features extracted from the convolutional layers are then used in a more classical Neural Network for classification.

2.2 Transfer Learning

Transfer learning consists of reusing a pre-trained model for a new problem. This technique arose with the rising complexity of neural networks over time. More complex networks require more data to converge. The idea of transfer learning is motivated by the observation that features extracted from a CNN are extremely generic, as they track borders and blobs, and they can therefore be reused in other tasks.
To reuse a network it is necessary to modify it and fit it to our needs. Usually, the classification layers are removed and replaced with a classification network that agrees with our output. This grafting procedure can also be done on the convolutional layers if needed. After grafting our network, it is necessary to decide which weights to train and which ones to freeze. The idea is, that the more we freeze the less specialized the network will be for our task, but it will be faster to train. On the other hand, the more the weights that we train the more specialized it will get, but it will be more computationally expensive and it will need more data to converge.

3 Data

The dataset contains around 70'000 images of actors and their ages when the photo was taken. A bit of data cleaning is necessary to exclude photos where the date of acquisition is not available or the ones that contain more than one person. Since we're dealing with transfer learning, we decided to work with IMAGENET1K [5] pre-trained networks.

4 Approach

The general pipeline can be described as follows:

1. Data cleaning (as described in the previous section)
2. Data augmentation: since we're going to work with deep and complex networks
3. Choose the network to work with, we tried several: ResNet [1] (with 18, 50, and 152 layers), MobileNet v3 [2] and RegNet (y128gf) [6]
4. Choose where to cut and graft our network

5. Choose which weights to train and which not.

6. Train and test

4.1 Preprocessing: Data Augmentation

The augmentation procedure is applied when a batch is loaded on each element of the batch. Since this procedure is subjected to randomness, the model will train on a slightly different dataset at each epoch.
The operations used are horizontal flipping, resized cropping, color jittering, and affine transformation. To perform such operations we used the `torchvision.transforms` [4] library, already containing a set of functions to perform the operations that we need. We built an augmentation pipeline with `Compose`, containing the aforementioned transformations with the following parameters:

- `RandomHorizontalFlip(p=0.5)`: The random flip is applied to an image with 50% of the times.

- `ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.1)`: This applies random transformations to the brightness, contrast, saturation, and hue of an image. Each transformation is bounded by a range of values specified (0.2 in this case), where 0.0 represents no change and 1.0 represents the maximum change. Although it will never happen to find a photo of a green person, the lighting condition of the room might change the hue of the skin of person in question, hence the slight color jittering.

- `RandomRotation(degrees=30)`: This brings the classic rotation and translation of the image. As for the color jittering, we decided not to rotate the images too much in order not to train the model in extreme cases (e.g. it is hard to get an image of an upside-down face).

4.2 Metrics and Validation

The task given is a regression problem. While we relied on L1 loss for the training, we opted for a couple of correlation coefficients for training and testing. We measured the correlation between the guessed ages and the ideal predictor the guesses always perfectly. The correlation coefficients used are Spearman's and Pearson's. Differently from Minkowski losses, they provide much clearer interpretability, due to their property of being scale invariant. Also, by using correlation coefficients, we're underlying our interest in guessing the strength and direction of the predicted labels versus the actual ones.

4.3 Optimizers

We used SGD and Adam, both optimizers behaved similarly during our tests and led to similar results in terms of final accuracy, resulting in no significant difference in performances. Both optimizers were used with learning rates of: 0.0001, 0.001, and 0.01, leading to a preference of a learning rate of 0.0001 since it had the most consistent results.

4.4 Approached Architectures

The pre-trained networks that we fine-tuned are ResNet with 18/50/152 layers, RegNet, and MobileNetv3. The choice of these networks was led by their performance on IMAGENET1K. The best-performing one happened to be ResNet with 18 layers having all weights unfrozen, allowing a complete fine-tuning of the network. We tried to graft two classification networks onto them, both of which are fully connected as shown in Table 1 and 2. We used the larger classifier for larger pre-trained networks such as ResNet with 50/152 layers and RegNet and the smaller one for smaller networks such as MobileNetv3 and ResNet with 18 layers. The need for such differentiation arose from the size of the output of each network, in order not to encode too harshly a large number of outputs. [3].
From the architecture tested we noticed a common behaviour, the networks with a lower number of convolutional layers led to faster convergence of results, achieving high scores in fewer epochs both with frozen and unfrozen. This can be traced back to the fact that the first layers of the networks usually learn more general features, while the last layers are more specialized at the task to solve, and since these networks were pre-trained on classification tasks of different nature, larger networks had a higher percentage of layers highly specialized, whilst smaller network allowed us to have more room to adapt to our task faster[7].
This observation led us to choose as our main network the ResNet18, which was the perfect compromise over all the factors aforementioned.

Table 1: Smaller grafted classification network

Layer	**Output Size**	**Parameters**
Linear-68	[-1, 512]	Depends on the network
GELU-69	[-1, 512]	0
Linear-70	[-1, 256]	131,328
GELU-71	[-1, 256]	0
Linear-72	[-1, 32]	8,224
GELU-73	[-1, 32]	0
Linear-74	[-1, 1]	33

Table 2: Larger grafted classification network

Layer	**Output Size**	**Parameters**
Linear-2-9	[-1, 2048]	Depends on the network
GELU-2-10	[-1, 2048]	–
Linear-2-11	[-1, 512]	1,049,088
GELU-2-12	[-1, 512]	–
Linear-2-13	[-1, 128]	65,664
GELU-2-14	[-1, 128]	–
Linear-2-15	[-1, 1]	129

5 Results

As mentioned in the previous section, the best-performing network is ResNet with 18 layers. Here we report the performance of the network having its convolutional layers' weights frozen and unfrozen.

5.1 Fine tuning with locked convolutional layers

As Figure 1 shows, the results are not that promising although the prediction very roughly follows a linear correlation as the Pearson coefficient states a value of 0.549. This behaviour can be explained by a network that has not been able to specialize itself in this task.

5.2 Fine tuning with unlocked convolutional layers

As shown in Figure 2 the results obtained by unfreezing every weight leads to much better results. This can be explained by the fact that the network is now able to specialize the convolutional layers also, at the cost of having enough samples, which is the case. The predictions now stick much closer to the actual labels, although the variance of such is still significant, on the widest point, the width is about a decade.

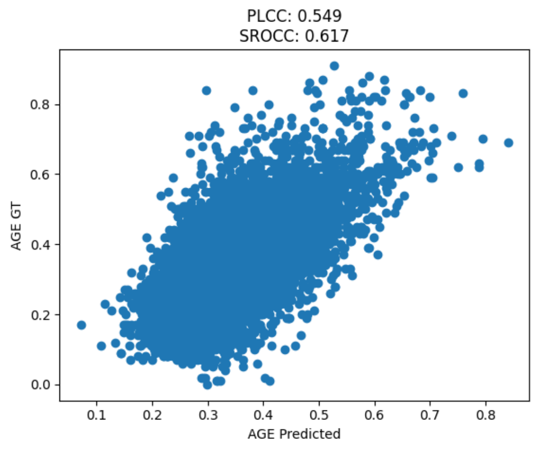

Figure 1: *Predicted age over actual age on the test set for frozen convolutional layers after 50 epochs*

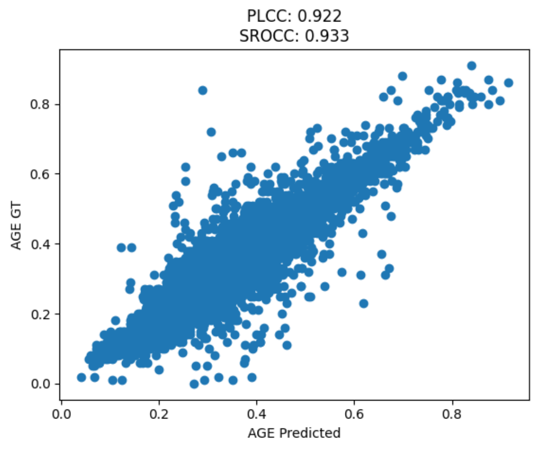

Figure 2: *Predicted age over actual age on the test set for unfrozen convolutional layers after 50 epochs*

6 Conclusions

The best-performing approach is the choice of ResNet with 18 layers having all of its weights unfrozen, with a rather small grafted classification network. The training time wasn't noticeably influenced by the choice of frozen and unfrozen solutions.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[2] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.

[3] Meta. Models and pre-trained weights — torchvision main documentation, 2017.

[4] Meta. Torchvision 0.11.0 transforms documentation, 2017.

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[6] Jing Xu, Yu Pan, Xinglin Pan, Steven Hoi, Zhang Yi, and Zenglin Xu. Regnet: Self-regulated network for image classification, 2021.

[7] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks?, 2014.