BEBENSEE BJOERN, 2019-21343

GROUP A:

A1. If the loss does not decrease and never manages to decrease as training goes on the learning rate may be too large. One can imagine that if the learning rate is too large the weight values will “bounce” around rather than converging. If the weights do not manage to “move” in a direction of a local minimum of the loss function, nothing is learned from the training samples and thus the loss function the loss will not decrease from the start.

A2. Convolutional neural networks can exploit certain characteristics specific to features seen in images such as locality. Pixels far away in an image are unlikely to be related and as such can be ignored for computation of local features. While a classical feedforward neural network will take into account all input pixels, a CNN takes only into account local pixels in a pre-defined window. Additionally, these local transformations (convolutions) share weights across the entire image meaning the number of parameters can be vastly reduced (in contrast: in feedforward neural networks the number of parameters grows with the image size).

A4. Even a different task may benefit from transfer learning as it can take advantage of the representations/embeddings learned from the first task (much like pre-training tasks in self-supervised learning). The weights learned in the first task already allow the network to extract and encode useful information in the representation and thus transfer learning is effective for different tasks that could benefit from these learned representations and extracted features as well.

A5. A neural network will typically overfit when it is trained for too long, has too many parameters or too little training data. One can imagine that when a neural network overfits it will place the decision boundary too “tight” to the training samples. That is, rather than learning a function that manages to generalize even to new samples, the network simply memorizes the training data. Due to this, the training loss will be low while the validation loss will be high and increase further as training continues.

A6. Non-linearities are necessary for the neural network to be able to learn non-linear functions (such as for instance XOR). Without a non-linear activation function the function learned by the network will always be a simple linear transformation of the inputs and thus the expressive power will be the same regardless of whether the neural network has a single, ten or a hundred layers.

GROUP B:

B2. Presumably these open-source pre-trained models are good at extracting useful representations from faces (as they work well for Europeans) but are not good at the actual identification/classification of faces. Thus, it should be possible to build upon these existing solutions and improve them for our use case by finetuning them on a

dataset of faces similar to the users in the app's target group. We could start by collecting a large amount of faces, e.g. Korean celebrities as these should be easy to acquire using a search API (e.g. Bing API, Naver API). It is important that we include a large variety of faces of the same person from different angles, lighting etc. so that the model can learn to recognize the same face while differentiating between different faces of different celebrities. After data collection is complete we should make sure that the collected data is correctly labelled and that there are no obvious mistakes in the training data (e.g. images of objects etc). Since we can not guarantee that the face is visible at all, we will need to further preprocess the data by cropping out faces and only keeping the images where faces where found. We can further augment our data by translations in the cropping process, changes in contrast, saturation, slight rotations, etc. Depending on the pre-trained model used, it may be beneficial or even necessary to dewarp the images (there are libraries and pre-trained models for this as well) such that we obtain an image that shows the face like it would look like from the front. We can now finetune the pretrained model on this dataset of Asian faces (or include other ethnicities as well if they will be in the app's target userbase!), perhaps freezing the first few layers of the pre-trained models or training them at a much lower training rate to ensure we do not "forget" the useful feature extractions. After validation, testing etc. this model can then be deployed in the app and finetuned on-device. We do this by adding a last classification layer on top of the feature extractions (in place of the previous classification layer used for celebrities) and train it on the user's face so that they can be recognized by the app.

B3. To train such a model we will first need a dataset of course. We can collect such a dataset from the internet, for instance by use of websites such as Wikipedia, IMDB and other databases that contain images of people along with their age (taking into account when the picture was taken relative to their birthdate). While it will be relatively easy to collect images of adults from these databases, they may be lacking images of children. We can collect further data from image search using e.g. the Bing API and searching for terms such as "6-year-old child". While collecting the data we should also make sure to include an even amount of both male and female training samples and a wide variety of ethnicities to reduce training bias. After collecting the data, we will need to make sure it is clean and the labels are mostly correct, removing any images that are obviously wrong or perhaps not of people at all.
We need to preprocess the data by cropping only the face (since our goal is to classify face images). We can further augment the dataset by rotations, mirroring images, cropping them differently (i.e. not always centering the face), changing contrast and saturation etc. Since the amount of collected training data may still not be large enough to train a large convolutional neural network model for this task, we may benefit from transfer learning as well. We could try using for instance an off-the-shelf facial recognition model that manages to extract good representations of faces from images that are useful for identification and add a classification or linear regression "head" to the model for finetuning for our age prediction task.
Although it may make sense to treat this task as a regression problem due to its numerical nature, we could also treat it as a classification problem where each class is an age bin, the size of which depends on the amount of granularity required in our

prediction (i.e. is it enough to know the person is 20~25 years old?). The disadvantage of this classification approach is that this does not directly encode how close different age groups are to one another while the regression approach inherently does.