Recognition of Facial Features with Deep Learning

(Homewrok project)

Bolla Gergő Levente
Schifferer András

In this homework project we choose the task of creating a neural network that recognises facial features from images. We managed to create this network with now ??% validation accuracy.

To create this project, we choose the FairFace dataset [1]. This dataset was well prepared for us since the faces were already extracted and the images were in a standard 256x256 pixel size. In order to prepare the dataset for the training we used embeddings since it makes a lot of things easier. We used FaceNet embeddings, it creates a mapping from face images to compact Euclidean space. Here the distances correspond to a measure of face similarity [2]. In this space face recognition and feature selection is much easier than using the standard images. We have a separate notebook for creating these embedding and saving them to google drive (we will provide a link to a drive so you can run it without downloading it since it takes a long time to create the embeddings). The embedding procedure as detailed in the paper [2] consists of first a rule based facial feature extraction. Only after transforming the faces into a canonical pose, so the eyes and edges of the faces always fall into similar domains, is the neural network based embedding step performed. After the embeddings are created, we loaded the labels from a csv file and onehot encoded them using Scikit-learn's preprocessing tool. After the embedding we created a separate loss function because the standard ones didn't prove to be useful (1. figure).

```
class MulticatLoss(keras.losses.Loss):
        def __init__(self):
            super(MulticatLoss, self).__init__()
            self.cce = keras.losses.CategoricalCrossentropy()

        def call(self, y_true, y_pred):
            return self.cce(y_true[...,0:9], y_pred[...,0:9])+\
                   self.cce(y_true[...,9:11],y_pred[...,9:11])+\
                   self.cce(y_true[...,11:18],y_pred[...,11:18])+\
                   self.cce(y_true[...,18:20], y_pred[...,18:20])
```

figure 1. The loss function

This loss function ensures that each feature receives the same treatment in terms of loss. For each featureblock one crossentropy is calculated, and they are added together, so all of them are minimized.

Next, we created a network and used Keras-tuner for hyperparameter optimization. We created different sized networks with different parameters and dropouts. The final network can see on figure 2.

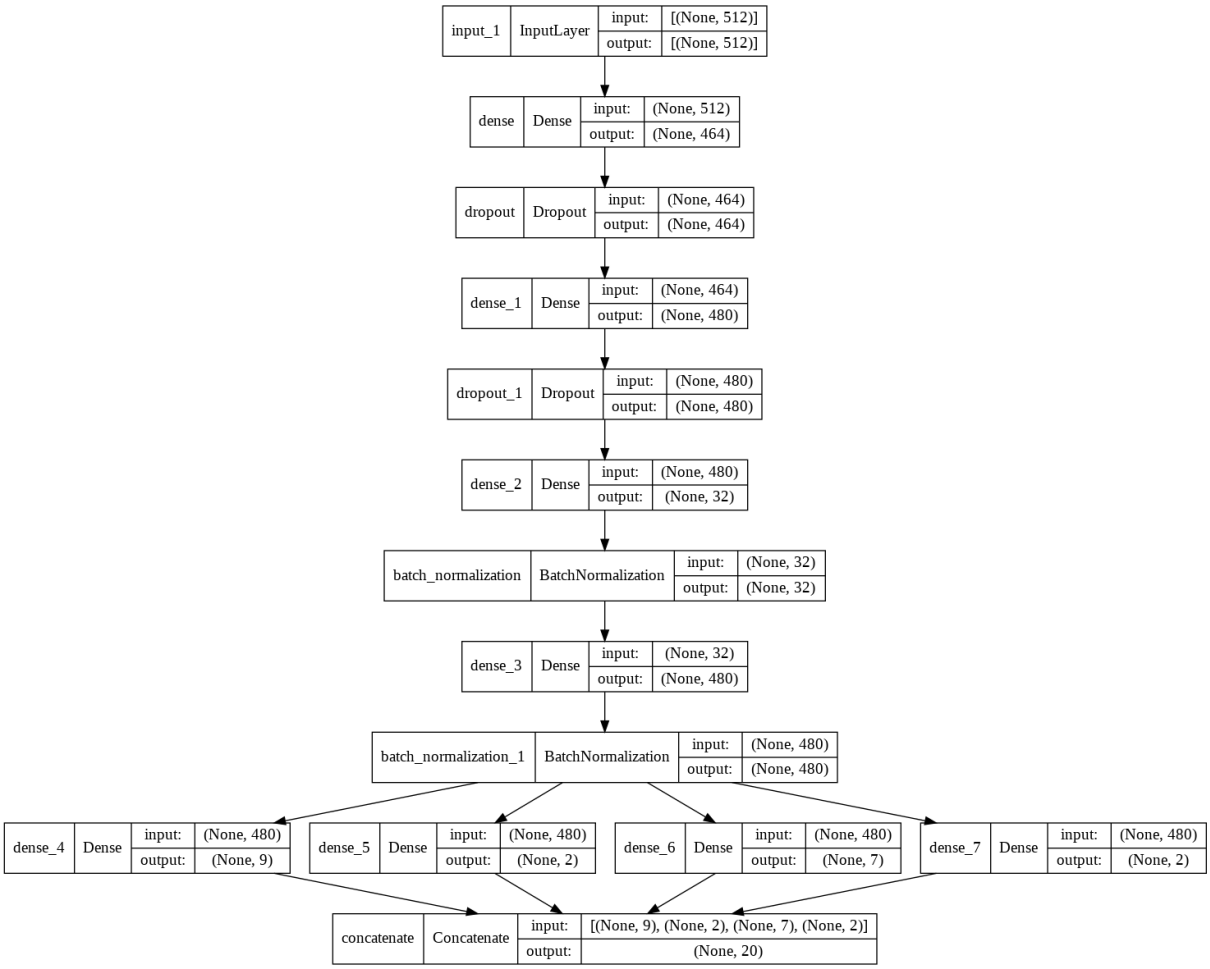

figure 2. The best neural network

After the optimization was done, we further trained the model and evaluated the results which you can see on figure 3.

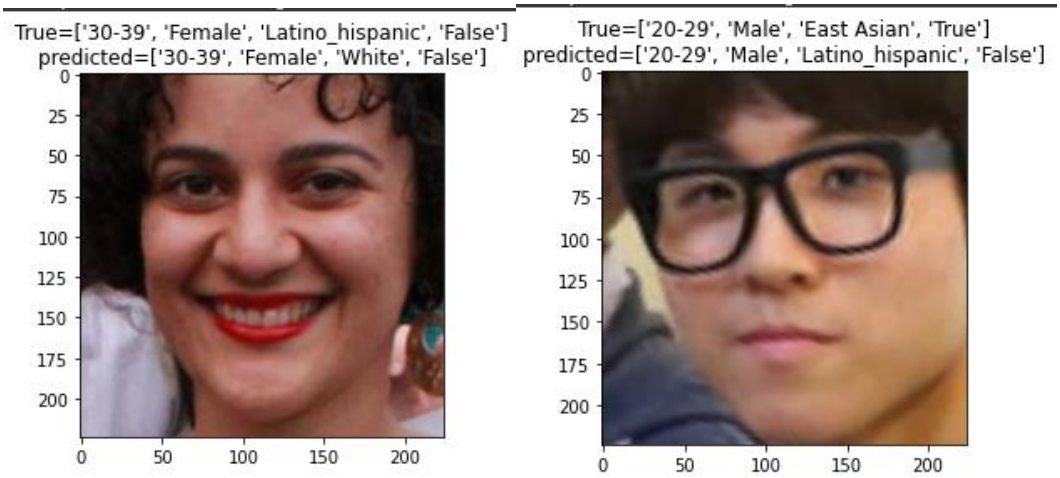

figure 3. Pictures with predicted labels

As you can see the model became pretty accurate at recognising these features from faces.

The results of the model can be best assessed with a confusion matrix. It is obvious from this, that even though in some aspects the dataset is very balanced, in others it is not so much so the model ends up relying mostly on statistics unfortunately. This is the most obvious in the age feature.

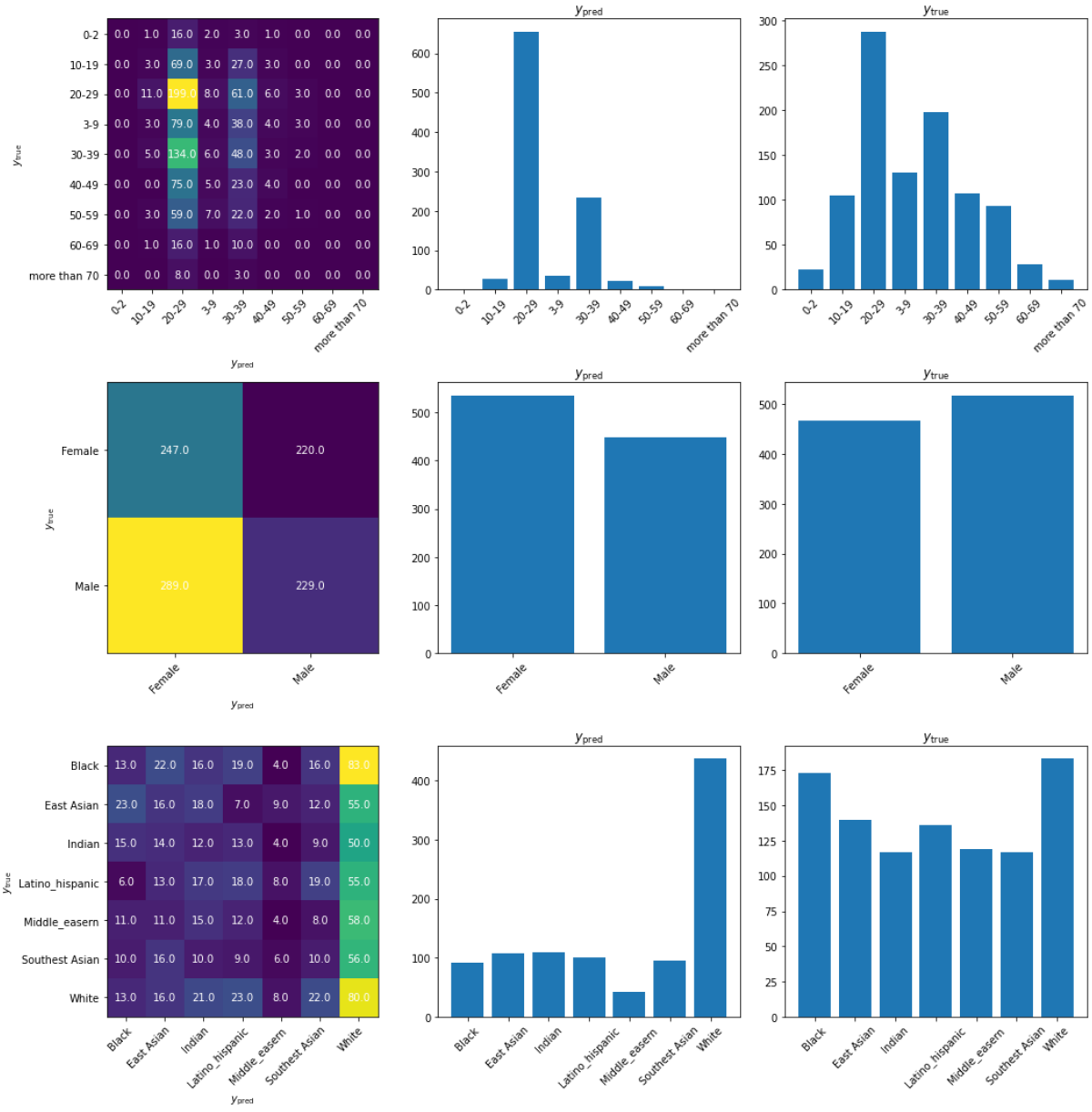

figure 4. Confusion matrix and data distribution.

References

1. Joo, Jungseock. FairFace: Face Attribute Dataset for Balanced Race, Gender, and Age. 2019. 2021. GitHub, https://github.com/joojs/fairface.
2. Schroff, Florian, et al. 'FaceNet: A Unified Embedding for Face Recognition and Clustering'. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015, pp. 815–23. arXiv.org, https://doi.org/10.1109/CVPR.2015.7298682.