Biometry and Applied Intelligence - Face Biometrics: Tasks 3, 4 and 5

Javier Muñoz Haro
Universidad Autónoma de Madrid
Ciudad Universitaria de Cantoblanco. 28049 Madrid
javier.munnozharo@estudiante.uam.es

Abstract

For this final laboratory, we are going to delve into the realm of feature embeddings in face biometrics by developing three tasks. In the first exercise, Task 3, we will analyse the provided database, DiveFace, and extract the embeddings using a face-specific feature extractor. Then, we will perform classification over those embedding for different classification-goal purposes (gender, ethnicity and demographic group classification). For the second exercise, Task 4, we will make use of the ethnicity and gender labels to train different "biased" classifiers and perform a cross-validation between them with data they have never seem before in order to see if different aspects from people's appearance affect the way neural networks learn. Finally, for Task 5, we will use different techniques found in self-supervised learning such as using contrastive learning, to test different biased and unbiased configurations over the dataset and see the learned features in low-dimensional spaces. Throughout this document we make several references to the code developed to solve these tasks. Said code is available in https://github.com/Javimh18/BAI_P3

1. Introduction

One of the most useful tools that machine learning has provided, are embeddings. An embedding is defined as a vector that holds a compressed and meaningful representation of an input space. Several Artificial Intelligence (AI) fields, such as Natural Language Processing (NLP), generative AI or biometrics make use of these representations for their own purposes. The edge that this tool provides rests on the ability of neural networks of extracting representative features from an input of any form (*e.g.* images, audio files or text), so if the neural network model is accurate enough, there is a high chance that the learned features contain essential compressed information. To extract embeddings the set-up is usually the same. First, we choose a pre-trained good enough model that may help us for the desired task (*e.g.* audio or image classification, sentiment analysis...) and some layers before the classification layer, we perform a cut in one of the dense layers, leaving the activations of those neurons as the output of the feature extractor. These activations will be the feature embeddings used for our own purpose task. We think that is also worth mentioning that working with embeddings give additional benefits in terms of computational costs, since we are not training or fine-tuning a big feature extractor, but using the obtained embeddings to train a smaller dense network with far less parameters.

In these lab sessions, we will perform a set of tasks using VGG-Face [4] as our feature extractor. The reason behind this model is that our dataset (DiveFace) is composed by images of different genders and ethnicity and VGG-Face is a model that is trained to detect representative features about different faces from diverse subjects, giving embeddings with rich information. Our goal then, will be to use the embeddings to analyse how neural networks learn to separate different classes in different tasks over a feature learned space.

2. Task 3

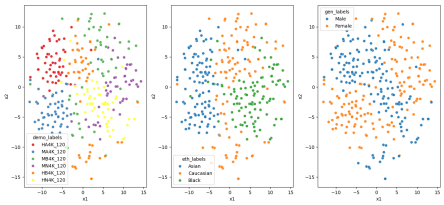
2.1. Extracting the Embeddings

Since our database is composed by raw images, the first thing we did was extracting the embeddings using VGG-FACE. Constrains were provided in the way the images were selected to obtain their corresponding embeddings, and the logic behind this is programmed consequently in this script.

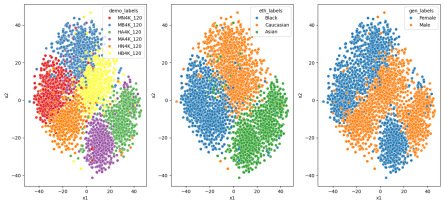
Once we obtained the embeddings, we performed the t-distributed Stochastic Neighbor Embedding (t-SNE) [2] method over them to visualize the data points in a reduced dimensional space. This is useful for several reasons, but the main one is that we can reduce the dimensionality of the embeddings to two or three dimensions and plot them onto a graph to see each class belonging to a different cluster over the compressed space.

As the exercise demands, we extracted 50 different embeddings, each from a random identity inside the DiveFace

dataset, after that we plotted the results that are available in figure 1a, but since there was a lot of sparsity, we extracted 500 embeddings from different identities and repeated the process in figure 1b. We can see that there is separability on them, which would be beneficial for a potential classification task. The code for obtaining these plots is available here.



(a) t-SNE for two dimensions, 50 data-points



(b) t-SNE for two dimensions, 500 data-points

Figure 1. Loss and accuracy for the ethnicity and gender classification.

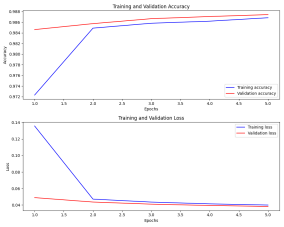
Also, we extracted the same data-point cloud but with three dimensions, to see if additional information was provided in the compressed space by adding an extra dimension. The code for this extraction is available here.

2.2. Training models for classification

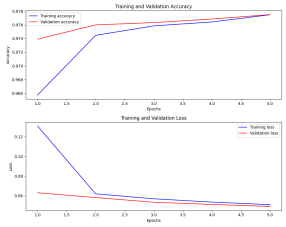
As we have seen in section 2.1, there is a high chance that the classes are separable in demographic groups, ethnicity and gender. This means that the features embeddings of VGG-Face contain useful compressed information about the features of the people's images from the dataset. To test this hypothesis, we trained a neural network classifier for classifying ethnicity and gender. The results are in figures 2a for the gender classifier and figure 2b for the ethnicity classifier. From both images it is clear that the model has no issue obtaining excellent results. This also means that the features used obtain useful and representative information about the images provided in the dataset. The code is available here.

3. Task 4

Now that we have established that the embeddings are of great use, we can proceed with more interesting tasks evaluating different aspects of the training procedure, such as bi-



(a) Training (blue) / Validation (red) loss and accuracy for gender classification throughout training for 5 epochs



(b) Training (blue) / Validation (red) loss and accuracy for ethnicity classification throughout training for 5 epochs

Figure 2. Loss and accuracy for the ethnicity and gender classification.

ased data and the generalization capabilities that neural networks achieve given a dataset. For this forth task, we trained the gender classification model, but with biased datasets. This means that we trained 3 gender models: the first one only with data from the asian ethnicity, the second one with only data from the black ethnicity and the last one with only data from the caucasian ethnicity. For each ethnicity, the training dataset had 1000 samples while the testing (evaluation) dataset had 500 samples. After the training process is performed, we made a cross validation comparison for each model and each ethnicity sub-set. The hypothesis that we dealt with while developing this task was to test if models can generalize well from different sources of data when a feature is common, in this case gender. Results can be observed in table 1 and the code developed for this task is here.

Ethnicity	Asian	Black	Caucasian
Model			
Asian	0.98	0.91	0.86
Black	0.94	**0.97**	0.91
Caucasian	0.91	0.89	**0.97**
All	**0.99**	0.96	**0.97**

Table 1. Gender classification for 3 different models. Each model has been trained in a subset of the original data, based on the ethnicity.

Observing the results, one of the things that caught our eye, was that skin color is not a relevant feature to detect the gender. For example, Asian and Caucasian people share skin tone, but the Asian detector fails considerably when detecting the gender of Caucasian people. Following this trend, we can see that it is precisely the model trained with black people the one with greater average accuracy between the three of them, supporting the hypothesis that colour is not an important feature for classifiers in order to classify gender. Finally, we trained a general classifier, with all the

data from different ethnicities, with best overall accuracy.

4. Task 5

For this final task we again took a look to the learned representation spaces that the model creates when trained over different datasets and loss functions. The set-up for this task was the same that in tasks 3 and 4. We trained the model using the extracted embeddings from the VGG-Face model and 3 different loss functions: softmax (also known as cross-entropy loss), contrastive loss and triplet loss. We also used a unbiased dataset and a biased one, to observe differences along the classification tasks in the learned space. In order to obtain the compressed features, we introduced a bottleneck layer between the layers and the output of the network. Let $W_o \in \mathbb{R}^{h \times o}$ be the weights of the neurons that go from the hidden layer to the output layer, if we introduce a layer such that $W_l \in \mathbb{R}^{h \times 2}$ and then the final layer has the weights $W_s \in \mathbb{R}^{o \times 2}$, then by performing the dot product $W_l W_s^T$ we would obtain a equivalent representation to W_o, with the advantage that we can look into the compressed space.

4.1. Learning with softmax loss

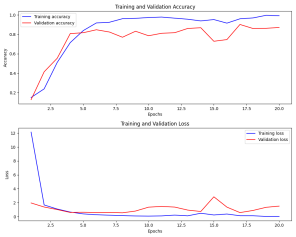
The softmax or cross entropy loss is defined in 1, and the main intuition is to highly penalize the predictions that are quite far from the original label.

$$CE(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right) \tag{1}$$

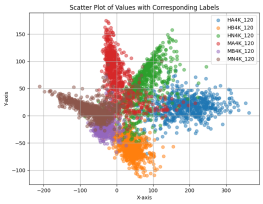
We trained a similar model to the task 3 and task 4, with mild changes on the architecture. The accuracy and loss over the epochs is shown in figure 3a, and the learned feature space is shown in figure 3b. We can see a mild over-fitting of the network, we tried to use dropout and initialize the neurons according to Xavier initialization, but we could not mitigate the over-fitting completely. This also happened with the rest of set-ups from this task. Also, we would like to emphasize the importance of the learning rate hyper-parameter in the optimizer, since we had to launch several runs with different values to obtain an "optimal" result (0.015). The implementation is accesible here.

4.2. Learning with contrastive loss

The contrastive loss was introduced by Le Cunn *et al.* in [1]. This approach was one of the pioneer work in the world of self-supervised learning. The loss function is defined in equation 2. We can see there are several terms: L_S is the loss of similar samples, L_D is the loss of dissimilar samples and D_W^i is computed using the L2-norm between the embeddings from the inputs $\overrightarrow{X_1^i}$ and $\overrightarrow{X_2^i}$ that are obtained



(a) Loss and accuracy over 20 epochs for demographic dataset.



(b) 2D embedded feature space separability between demographic groups.

Figure 3. Loss and accuracy for the ethnicity and gender classification using softmax loss.

from an identical feature extractor G_W, that in this case our neural network, as shown in equation 3.

$$\mathcal{L}(W, Y, \overrightarrow{X_1}, \overrightarrow{X_2}) = \sum_{i=1}^{N} (1 - Y) L_S(D_W^i) + Y L_D(D_W^i) \tag{2}$$

$$D_W(\overrightarrow{X_1}, \overrightarrow{X_2}) = \| G_W(\overrightarrow{X_1}) - G_W(\overrightarrow{X_2}) \|_2 \tag{3}$$

The design of the loss for the similar and dissimilar embeddings must be done accordingly to a simple goal: pull similar instances closer and push away dissimilar samples in the embedding feature space. To do so, in the L_D term, a margin m is introduced, so that the dissimilar instances are forced to be m distance or more away from each other, while bringing closer the similar instances. The final form of the loss is defined in equation 4.

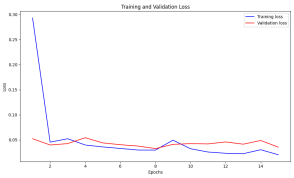
$$\begin{aligned} \mathcal{L}(W, Y, \overrightarrow{X_1}, \overrightarrow{X_2}) = \sum_{i=1}^{N} & \frac{1}{2} (1 - Y) (D_W^i)^2 \\ & + (Y) \frac{1}{2} max(0, m - D_W^i)^2 \end{aligned} \tag{4}$$

Using this loss with a margin value m = 0.5, we trained the model for a total of 15 epochs with 17.000 samples in train, 3.000 in validation. We obtained the results shown in figure 4b for the feature embedding space for a total of 1000 embeddings in the test subset.

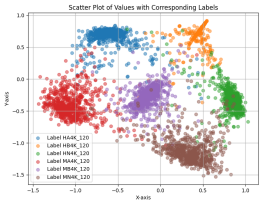
One of the things that caught our eye was that the distance between the clusters is similar to the margin m = 0.5, that can be taken as evidence that the separation has been done properly. The code is accesible here

4.3. Learning with triplet loss

Another way of training in a self-supervised way is by using the triplet loss function, introduced in [3]. The intuition is similar to the contrastive loss in section 4.2; we want the intra-class variability to be minimized, while the inter-class

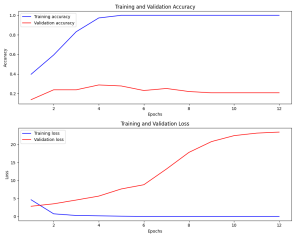


(a) Contrastive loss over 15 epochs for demographic dataset.

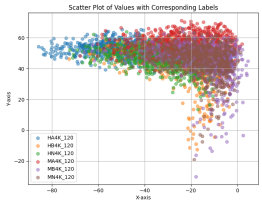


(b) 2D embedded feature space separability between demographic groups.

Figure 4. Loss and accuracy for the ethnicity and gender classification using contrastive loss.



(a) Softmax loss over 12 epochs for demographic dataset.



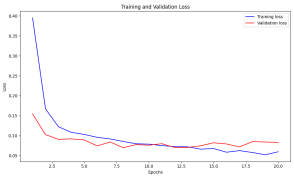
(b) 2D embedded feature space separability between demographic groups.

Figure 6. Loss and accuracy for the ethnicity and gender classification using triplet loss.

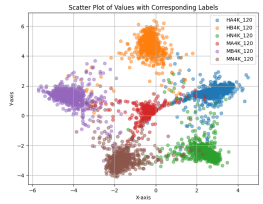
variability to be maximized. To do this, triplet loss uses an anchor, that is the embeddings that the model is going to use to learn, and then a positive embedding, that belongs to the same class as the anchor, and a negative embedding that belongs to another class.

$$\mathcal{L}(A, P, N) = \max\{0, \|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha\} \tag{5}$$

We trained the model for 20 epochs, with a learning rate of 0.015 using Adam optimizer. We also added a dropout layer since we observed that the model was quite prone to over-fitting. Results are available in figure 5, where we see that clusters are a bit smaller, indicating low intra-class variability, and the separation between classes is bigger, indicating higher inter-class variability. The implementation is accesible here.



(a) Contrastive loss over 15 epochs for demographic dataset.



(b) 2D embedded feature space separability between demographic groups.

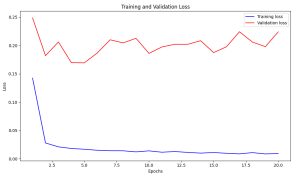
Figure 5. Loss and accuracy for the ethnicity and gender classification using triplet loss.

4.4. Biased learning

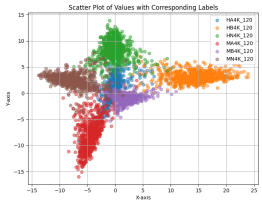
For the last part of this section we are going to test how robust are the different losses when put under a biased dataset. In our case, we used the demographic labels to perform such task, being the asian male (HA4K_120) the label which our dataset would be biased towards.

In the case of the softmax loss function, we see in figure 6 that results are quite poor, because the model has seen so little of the rest of the classes, it does not separate well their features, resulting in a mesh of datapoints.

In the case of the triplet loss, it is observable that there is a clearer and greater separation between classes. We attribute this to the fact that contrastive learning does not model depending on the output's logits, but instead, the separation of the learned feature space. We could say that, in a way, contrastive learning is "inmune" to biased datasets or class imbalance.



(a) Softmax loss over 12 epochs for demographic dataset.



(b) 2D embedded feature space separability between demographic groups.

Figure 7. Loss and accuracy for the ethnicity and gender classification using triplet loss with biased data.

References

[1] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pages 1735–1742, 2006. 3

[2] Geoffrey E. Hinton and Sam T. Roweis. Stochastic neighbor embedding. In *Neural Information Processing Systems*, 2002. 1

[3] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015. 3

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 1