Transfer Learning for Facial Attributes Prediction and Clustering

Luca Anzalone, 0522500638, **Simone Faiella**, 0522500693, **Marialuisa Trerè**, 0522500652

***Abstract*—The recognition of facial attributes, in the field of machine learning, is still today a topic of study and research. In this paper we propose not only a trained model (using transfer learning) to recognize specific attributes of a face, but also an unsupervised clustering model that divides and groups faces based on their characteristics. Furthermore, we show how clusters can be evaluated by a compact summary of them, and how Deep Learning models should be properly trained for attribute prediction tasks.**

***Index Terms*—attribute clustering, k-means, face attributes, transfer learning, cluster summary**

I. Introduction

Recognizing and grouping facial attributes is very important, and can be used in different applications, such as face verification and identification. This task became very challenging when the subject is not collaborative. In fact, in a collaborative scenario many face variations can be eliminated or greatly reduced: such as illumination and pose. Instead, in an uncooperative scenario, in which the user can be not aware of being captured and analyzed, even detecting the subject face becomes very hard due to complex variations that alters its appearance significantly.

Common face variations encountered in face-related tasks are:

- **Occlusions**: occurs when objects (e.g. hat, eyeglasses, etc) covers relevant parts of the face, eventually hiding underneath features.
- **Pose**: occurs when a face shows high-degree rotations in terms of roll, yaw, pitch.
- **Illumination**: occurs when a face is affected by a too high or too low brightness.
- **Expressions**: occurs when a change in expression alters the face features, thus reducing the change of a correct analysis.

So, a face captured in unconstrained environments must be normalized before being further analyzed.
Some approaches for pose estimation are based on a mixture of trees model [1], while other are able to detect even occlusions by performing landmarks estimation [2]. But the most effective and robust approaches rely on neural networks (NN) especially convolutional neural networks (CNNs) [3]–[5].
As discussed previously, our approach rely on CNNs. In this way the model can learn robust features (compared to hand-made features like HOG, Haar, LBP), allowing it to infer facial attributes.
In this article we propose a pipeline for detecting and grouping faces, in respect to the discovered facial attributes. In particular we propose:

1) A model heavily based on the Transfer Learning approach, that estimates facial attributes; along with some training insights.
2) According to the detected attributes, we evaluate different clustering methods in order to discover and visualize the best grouping technique.

II. Related works

Although the recognition of characteristics and the relative grouping using Clustering techniques have been extensively analyzed in the literature, the clustering of facial images is a less discussed topic. Since there is no universally accepted facial representation or distance metric, the clustering results depends on: not only on the choice of the clustering algorithm, but also on the quality of the face representation and the adopted metric. However, it should be emphasized that often the analyzed papers, focus only on one of the two main aspects that compose our work. In fact, they are more concerned with the development of a model or clustering techniques on a small number of characteristics.

A. Attribute prediction

A work strongly related to the problem we faced was conducted by Ziwei Liu et al. [3], however it is limited exclusively to the creation of a model, not dealing in any way with clustering (the researchers only discover related attributes).
More in details, they have created a model for predicting facial attributes, cascading two CNNs: LNet and ANet; each of them pre-trained in a different way: the former on massive categories of general objects (for face localization), and the latter on large number of facial identities for attribute prediction. Then, either are trained and fine-tuned jointly with attributes tags.
Pre-training ANet on face identity allows the model to manage complex face variations thanks to the learnt face features.
Another possible way to deal with this problem is to learn a discriminative face representation. Researches in [6], [7], proposed a model that synthesize a face into a compact representation, called face embedding, that, being properly trained, is able to take apart faces belonging to the same subject (identity).
During the training process, the model implicitly learns enough features to distinguish face identities (the embeddings related to the same identity will lie on the same hyperplane). However the resulting face embedding is hard to interpret because it hides the learnt facial features, consequently losing any relations among attributes.
Thus, the embedding-based approaches are extremely good at

maximizing the performance of a single feature; in this case the feature is the face identity. Indeed these are one of the most effective way of performing accurate face recognition tasks (which are based on a single high-level concept: identity).

B. Clustering methods

For our purposes we compared three clustering techniques: K-means, Agglomerative Clustering, and DBSCAN.
In this section we briefly discuss about their functionality.
The K-Means algorithm [8] clusters data by trying to separate samples in n groups of equal variance, minimizing the inertia criterion (the average squared distance between points in the same cluster). This algorithm requires the number of clusters to be specified as an input parameter. It scales well to large number of samples and has been used across a large range of applicative domains.
The Agglomerative Clustering approach falls within the Hierarchical clustering family. Hierarchical methods works by building nested clusters, obtained through merging or splitting underneath clusters. This nesting process stops when a single cluster is constructed; so the hierarchy of clusters is usually represented as a tree or dendrogram.
Differently from the other two, the DBSCAN algorithm is able to discover clusters of arbitrary shape; so, it doesn't require the number of clusters as input parameter. Moreover, DBSCAN is designed to require a minimal knowledge of the domain data, together with a good efficiency on large databases.
Otto et al. [9] faced the problem of clustering millions of faces into thousands clusters of related identities. They proposed an approximate Rank-Order clustering algorithm that performs better than popular clustering algorithms (k-Means and Spectral) by achieving a better accuracy and run-time complexity.
Finally, among the various works analyzed, another interesting evaluation was carried out on the work done by Rosebrock [10]. In this case an algorithm has been developed which extracts an array of 128 real numbers from each face and creates clusters with these data.
Each cluster contains a set of faces with similar facial features. Obviously, using a clustering approach based on DBSCAN, the number of the created clusters depends on the algorithm, which is structured to return an optimal number of clusters.
Even in this case it doesnt fit our needs perfectly, as another goal that has been set is the possibility of independently choosing the number of clusters to be displayed.

C. Transfer learning

As the research goes further, the complexity of the ML tasks increase. The resulting model architectures are big too, being slow to train. This is true, especially for Convolutional Neural Networks [11] that requires a huge amount of data and computational power.
Thanks to the ImageNet classification challenge [12], many effective models have been trained (like the groundbreaking AlexNet [11]). It turned out that models like VGG [13], InceptionNet [14], and ResNet [15] are very good for solving general ML tasks.
Thus, the Transfer Learning [16] techniques allows to recycle a model architecture in order to perform a different task from the one it was developed for.
Common tasks are related to features extraction and fine-tuning. In particular we utilize Transfer Learning to fine-tune our model.

III. Our approach

In this section, we give more details (even technical) about the proposed detection-clustering pipeline.

Basically, the entire workflow consists of three parts:

- **Attributes inference**: in which we fine-tune a pre-trained model on 37 facial attributes taken from the CelebA [3] dataset.
- **Attributes clustering**: the labels predicted by the proposed model are given, as input, to a commonly available clustering algorithm (such as K-Means) that computes the grouping of the input faces according to a criterion of closeness.
- **Visualization and analysis**: we show that by simply computing the occurrences of the attributes (of the given clusters), these are enough to evaluate quantitatively the goodness of the resulting clustering. Moreover, we provide an even more concise way to graphically evaluate clusters.

A. Model

Our framework of choice is Keras [17]. This framework allows to build, train and evaluate models with little effort. In addition, Keras offers a variety of ready-to-use pre-trained models.

- **CelebA**: All experiments and studies have been conducted on the CelebA dataset. Unlike other datasets of faces (like LFW [18]), CelebA is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations.
  This dataset is very challenging and, at the same time, appealing because it contains face-images covering a large amount of pose, expression, age, and occlusion variations.
  We found out that CelebA has heavily imbalanced attributes (figure 1): more than a third of attributes are extremely rare (with frequencies below 10%), and only a couple of them are very common (by occurring more than 70% of the times).
  This imbalance has pointed out the weaknesses of widely adopted loss functions, when training a model on rare-occurring instances.

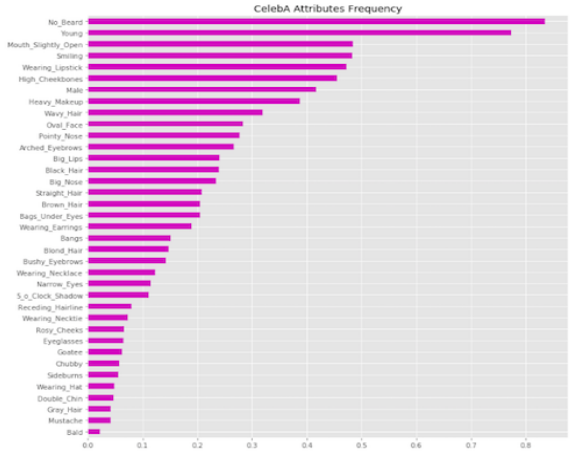

Fig. 1. Frequencies of CelebA attributes: the chart clearly shows the discrepacy about the occurrence of the attributes.

- **Transfer Learning**: We employ the Transfer Learning approach to fine-tune our model architecture, in order to achieve a faster training convergence.
  Our baseline model is represented by the MobileNetV2 [19] architecture, but without the top classification layers (figure 2).

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Fig. 2. MobileNetV2: Each line describes a sequence of 1 or more identical (modulo stride) layers, repeated n times.

Our top layers (figure 3), simply consists of a Fully Connected layer, followed by a Batch Normalization operation before the final multi-label Dense layer that outputs the facial attributes of the input sample.
So, the model outputs a binary 37-d vector (we decided to drop three attributes: attractiveness, pale skin and blurry).

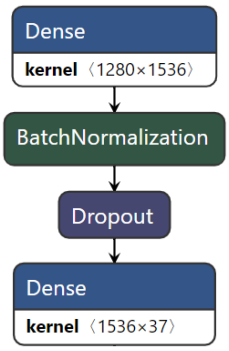

Fig. 3. Top Layers: A first Fully-Connected (Dense) Layer with 1536 neurons, normalized by Batch-Normalization and regularized by applying a dropout on 30% of the units. The last Dense layer outputs the labels for every attribute (thus, requires 37 neurons *sigmoid*-activated).

- **Data Augmentation**: In order to let the model generalize better, we use a common technique called *Data Augmentation*. This method consists of augmenting the training images by applying some random modifications to them. In particular we apply the following augmentations:
  - Rotation: the image is rotated by a maximum of 20 degrees.
  - Shift: a translation (shift) on either width and height is applied, with a factor of 0.2.
  - Shear: a random distortion is applied with an effect of 0.2.
  - Zoom: the image is magnified by a maximum of 20% of their total dimension.
  - Flipping: only horizontal flipping is applied to images.

  In this way the number of training instances can be effectively increased: the model will see slightly different training samples during each epoch.
- **Loss Issue**: We discovered, surprisingly, that common loss function like mean absolute error (MAE) and mean squared error (MSE), if used as the training objective, fails completely in their aim.
  These kind of loss function are not suitable for sparse binary multi-labeled data, when the position of a single bit is informative: different vectors but with the same amount of 1s (e.g. [0, 0, 1, 0] and [1, 0, 0, 0], versus [0, 0, 1, 0] and [0, 1, 0, 0]), produces the exactly same loss. Training a model with these loss function, leads to a completely dumb model: it predicts an array of 0s for whatever input instance regarding the belonging attributes. This scenario gets worse when we deal with rare (in this case sparse) features.
  For example: if the model has to predict whether a face is young or not, and that feature is 1 for only 5% of the instances, a prediction of 0 (for every test instance) results in an accuracy of 95%. But the truth is that the model has learned nothing.
  As discussed earlier, relying only on accuracy can be misleading (a confusion matrix will tell more about these kind of phenomena). In fact, we double-check the accuracy of our model with: qualitative results, and misprediction ratio.

Finally, a good loss function (not only for our purpose) should understand the difference between two samples. Cosine Proximity is good at this (figure 4).

$$\mathcal{L} = -\frac{\mathbf{y} \cdot \hat{\mathbf{y}}}{||\mathbf{y}||_2 \cdot ||\hat{\mathbf{y}}||_2} = -\frac{\sum_{i=1}^{n} y^{(i)} \cdot \hat{y}^{(i)}}{\sqrt{\sum_{i=1}^{n} \left(y^{(i)}\right)^2} \cdot \sqrt{\sum_{i=1}^{n} \left(\hat{y}^{(i)}\right)^2}}$$

Fig. 4. Cosine Proximity equation: the key ability of this loss function is to distinguish every combination of 1s (attributes) in a binary array.

It treats a binary array as a vector in a multidimensional space. The differences (or similarities) between the true and predicted labels are expressed as the $\arccos$ of the angle between them. When two vectors are orthogonal (theres a 90 angle between them), means that they are completely different (the loss value is at its maximum). Instead, when two vectors overlap (the angle is 0), they are completely the same (the loss value is at its minimum). The training process with cosine proximity is effective: it allows the model to effectively learn patterns, utilized to infer the facial attributes.

- **Training**: We divide the CelebA dataset into three partitions: *training*, *validation*, and *testing*; according to the partitioning suggested by the CelebA authors. For training, we feed the model with the entire training partition ($160k$ samples resized to 224×224 and augmented) and validate each epoch on $20k$ of samples (validation-set). We train the model with a batch size of 64 images (due to resource limitation), and let the AdaDelta [20] optimizer minimize the cosine proximity loss function. We choose AdaDelta as optimizer, because it requires less hyperparameter-tuning. Moreover, AdaDelta has a few interesting features:
  - It's able to adapt the learning rate according to a moving window of gradient updates, preventing to stop learning after many iterations.
  - It converges faster: thanks to the adaptive learning, it's able to accelerate when the direction is promising and to slow down when the loss starts to get worse.
- **Results**: Before focusing on the MobileNetV2 architecture, we have tried different model architectures even with 10x more parameters (our model has only 4.3M of parameters); each model achieves roughly the same accuracy, but at the cost of a much slower training.
  Our model achieves 90.95% testing accuracy (table I) and trains only in few epochs.

B. Clustering

In this work we face the problem of grouping facial attributes, according to the following requirements:

- Choose the number of clusters (no fixed values);
- Eventually, select a subset of the available attributes;
- Graphically visualize the resulting clustering;
- Synthesize a cluster into one face, that is the cluster's *eigenface*.

So, the target of the clustering method is to group together instances having similar (or exactly) attributes.
According to these requirements, we evaluate the goodness of different clustering algorithms by computing the *silhouette score* of each clustering. Furthermore, we give graphical insights by: attributes-occurrence plot, and the cluster's eigenface.
Considering that DBSCAN discovers the optimal number of clusters (for a given set of data), for a fair evaluation, we tested the performance of all the methods on the same number of clusters (the one discovered by DBSCAN), in order to analyze which one is the best (figure 5).
Subsequently, we compare the performance of both K-Means and Agglomerative Clustering on a variable number of clusters (figure 5).
In particular, as regards Agglomerative Clustering, various input parameters are analyzed to optimize performance. In fact, it process by merging clusters together according to a linkage criteria:

- **Ward** minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach and in this sense is similar to the K-Means objective function but tackled with an agglomerative hierarchical approach.
- **Maximum or complete linkage** minimizes the maximum distance between observations of pairs of clusters.
- **Average linkage** minimizes the average of the distances between all observations of pairs of clusters.
- **Single linkage** minimizes the distance between the closest observations of pairs of clusters.

Also, the linkage criteria determines the metric used for the merge strategy: The latter can be euclidean, l1, l2, manhattan, or cosine. If linkage is ward, only euclidean is accepted.
The best combination of metric and linkage criteria is "complete" and "manhattan", but despite this choice, the algorithm's performance is still inferior to that of K-Means. Agglomerative Clustering reaches a silhouette score of 0.73, while K-Means achieves a value of 0.83.
Despite the better performances of DBSCAN, with a silhouette score of 0.87, we have decided to use K-means, since, as can be evaluated from the charts, it prove to be the best among the algorithms that allow you to choose the number of clusters.

More in detail, the K-Means algorithm divides a set of N samples X into K disjoint clusters C, each described by the mean μ_j of the samples in the cluster. The means are commonly called the cluster centroids; they are not, in general, points from X, although they live in the same space.
The K-Means algorithm aims to choose centroids that minimize the inertia, or within-cluster sum-of-squares criterion:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2) \tag{1}$$

In the next section, the results of these evaluations will be further analyzed, showing in particular the frequency of occurrences of facial attributes within each cluster. Moreover, we study how clustering change by weighting the input attributes.

TABLE I
PERFORMANCE COMPARISON OF ATTRIBUTE PREDICTION MODELS

	5 Shadow	Arched Eyebrows	Bags Under Eyes	Bald	Bangs	Big Lips	Big Nose	Black Hair	Blond Hair	Brown Hair	Bushy Eyebrows	Chubby	Double Chin	Eyeglasses	Goatee	Gray Hair	Heavy Makeup	High Cheekbones	Male
Our model	95	81	85	99	96	71	84	90	96	89	92	96	96	99	97	98	91	87	98
LNets+ANet	91	79	79	98	95	68	78	88	95	80	90	91	92	99	95	97	90	87	98

	Mouth Slightly Open	Mustache	Narrow Eyes	No Beard	Oval Face	Pointy Nose	Receding Hairline	Rosy Checks	Sideburns	Smiling	Straight Hair	Wavy Hair	Wearing Earrings	Wearing Hat	Wearing Lipstick	Wearing Necklace	Wearing Necktie	Young	**Average**
Our model	94	97	87	96	76	76	93	95	98	93	83	82	90	99	91	88	97	88	**91**
LNets+ANet	92	95	81	95	66	72	89	90	96	92	73	80	82	99	93	71	93	87	**87**

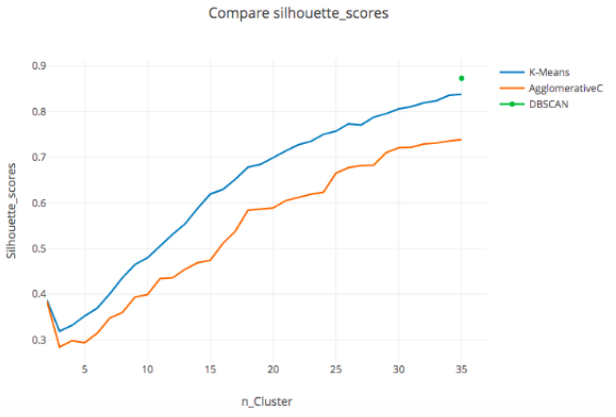

Fig. 5. Comparison of clustering algorithms: as we can see K-Means performs always better than Agglomerative Clustering.

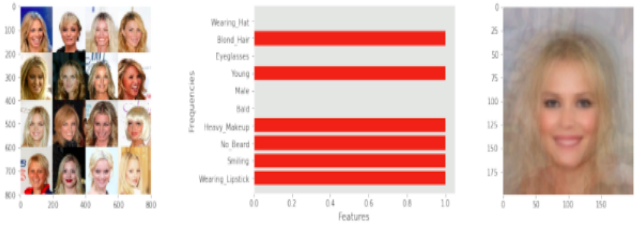

Fig. 6. Clustering results: the cluster (left), the attributes-occurence chart (middle), and the cluster's eigenface (right).

IV. RESULTS

Following the clustering process, the results obtained by each cluster are analyzed, both in terms of characteristics and in terms of the corresponding eigenface.
The eigenfaces are computed (by standard dimensionality reduction methods, like PCA) in order to obtain an exhaustive and meaningful representation of each cluster.
Hand-checking the attributes of the faces in a given cluster is tidy and error-prone, especially for clusters with a large amount of faces.

Regarding this, we show two simple way to summarize a cluster:

1) By simply showing the chart of occurrences of the facial attributes within a given cluster, is possible to understand the frequency of each attributes.
   With a chart like the one in figure 6, is possible to determine: (1) what are the noisy attributes (the ones with a low occurrence) - these outlier attributes are some sort of mistakes, made by the model and/or by the clustering method, (2) the prominent attributes (the ones with a frequency close to 1.
2) An Eigenface can be seen as a cluster representation, being representative and compact (its only one image).
   From a given cluster we generate its eigenface by simply performing Principal Component Analysis (PCA) [21] on vectors obtained by flattening the images belonging to that cluster.
   The resulting eigenface is a face characterized by the prominent attributes of the given cluster. So, by simply observing the cluster eigenface is possible to determine what are the relevant attributes of that cluster.

Analyzing the results. In order to further improve the clustering process, a weight criterion is adopted in relation to the most frequent characteristics, or alternatively, to the less

frequent characteristics.

Fig. 7. 1. Clusters without weights 2. Clusters with weights

Thanks to this expedient, it is possible to drastically reduce the quantity of features that are only partially present within the cluster. The chart shown, in fact, describes the results following this operation.

REFERENCES

[1] X. Zhu and D. Ramann, "Face detection, pose estimation, and landmark localization in the wild," Dept. of Computer Science, University of California, Irvine, 2012.

[2] X. P. Burgos-Artizzu, P. Perona, and P. Dollar, "Robust face landmark estimation under occlusion," The IEEE International Conference on Computer Vision (ICCV), 2013.

[3] Z. Liu, P. Luo, X. Wang, and X. Tang, "Large-scale celebfaces attributes (celeba) dataset," Multimedia Laboratory, The Chinese University of Hong Kong, 2015.

[4] R. Ranjan, V. M. Patel, and R. Chellappa, "Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," IEEE, 2017.

[5] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," IEEE, 2016.

[6] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," Google Inc., 2015.

[7] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," Georgia Institute of Technology,Carnegie Mellon University and Sun Yat-Sen University, 2017.

[8] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," 2006.

[9] C. Otto, D. Wang, and K. Jain, "Clustering millions of faces by identity," IEEE, 2016.

[10] A. Rosebrock, "Face clustering with python," 2018.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," University of Toronto, 2012.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," 2015.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[16] Link: "Http://cs231n.github.io/transfer-learning/tf."

[17] F. Chollet *et al.*, "Keras(2015)," 2017.

[18] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," 2008.

[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[20] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[21] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," 1987.