Face Deformation Measure And Expression Recognition Using Dlib's Facial Landmarks

Alberto Baldrati
alberto.baldrati@stud.unifi.it

Giovanni Berti
giovanni.berti2@stud.unifi.it

Abstract

In this work we describe how we exploited dlib's facial landmark recognition capabilities in order to measure face deformation and perform an expression recognition task. We implemented multiple approaches, based on supervised and weakly-supervised metric learning, neural networks, and geodesic distances on a Riemannian manifold computed on a transformation of the detected landmarks. To train the metric learning and neural networks models we built a small dataset made of eight facial expressions for each subject.

1. Introduction

In this work we address the measuring of the face deformation and the recognition of expressions. These tasks are intended to be executed in a real-time environment in order to give an online feedback to the user on its current progresses (e.g. to aid patients to correctly train facial muscles after a trauma).

Unfortunately because no facial expressions dataset was available a new one had to be created. For the sake of simplicity we only considered eight facial expressions, including a *neutral* expression used to normalize the other seven. At the end of the building process our dataset amounted to **328** photos.

Because of the dimensionality [1] issues that would arise with an image-based approach, all the proposed methods do not work directly on images, but rather they make use of facial landmarks extracted with `dlib`. This landmarks are extracted in the form of **68** (x, y)-coordinates that summarize the facial structure of the face and then are flattened into a **136**-d landmark vector.

All of our models are trained to output the distance between an online image (taken for example from a webcam) and an expression chosen by the user. The first approach we employed is based on supervised and weakly supervised metric learning [4] [5] [6]. The supervised model takes as input a vector of extracted landmarks and a label corresponding to the facial expression depicted in the original image. The weakly supervised model accepts as input pairs of landmark vectors with a label of $+1$ if they are similar and a label of -1 if they are dissimilar.

The second approach we propose is based on the use of neural networks. We built and trained a fully connected network which operates on the extracted landmarks. Due to the limited size of our dataset we used a fully connected network on landmarks instead of a convolutional neural network on images.

Finally we experimented with a static metric, which does not need training, based on geodesic distances on a Riemannian manifold after a transformation of the detected landmarks.

2. Dlib's Facial Landmarks

Dlib's facial landmarks make the basis of all our implementations since all of our trained models work directly on these landmarks.

Dlib's landmarks detector try to estimate the position of mouth, eyebrows, eyes, nose and jaws. Such detector is an implementation of *One Millisecond Face Alignment with an Ensemble of Regression Trees* [7], this method starts by using:

- A training set of labeled facial landmarks on an image. These images are manually labeled, specifying (x, y)-coordinates of regions surrounding each facial structure.
- *Priors*, of more specifically, the *probability on distance* between pairs of input pixels.

Given this training data, an ensemble of regression trees are trained to estimate the facial landmark positions directly from the *pixel intensities themselves* (i.e., no "feature extraction" is taking place). The end result is a facial landmarks detector that can be used to detect facial landmarks in real-time with high quality predictions.

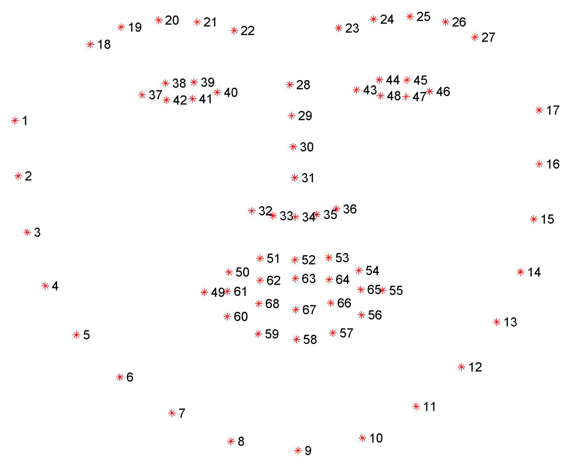


Figure 1. dlib's facial landmarks

The dlib's facial landmark detector is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face [8]. The landmarks extracted from dlib can be visualized in image figure 1.

The landmarks are expressed in term of image coordinates, which means that they have poor invariance proprieties. In order to have a more robust descriptor we perform some preprocessing steps:

- rotation of the landmarks around the tip of the nose to ensure that the line of the eyes is horizontal: this step achieves **rotational invariance**
- translation of the landmarks to ensure that the they are zero centered around both x and y axes: this step achieves **translational invariance**
- rescale of the landmarks to ensure that they have values between zero and one: this step achieves **scale invariance**

At the end of this three steps we obtain normalized landmarks which have values in the canonical square $[0, 1]^2$. As the last step of the normalization process we subtract the neuter expression landmarks of the same subject to the landmarks obtained from the previous three steps, so that the final landmarks are a normalized offset of the actual expression from the neutral one. This last step tries to achieve **facial geometry invariance**.

3. Dataset

As stated above due to the unavailability of a facial expressions dataset a new one had to be collected. We have considered eight facial expressions, including the neutral expression used to normalize the others. Note that the preprocessing pipeline we developed is expression independent, meaning that it can handle a expressions of variable number and kind (only the neutral expression is strictly required for the normalization purposes we outlined above). Here we report the expressions we included in our dataset, as shown in figure 2:

(a) *neutro*: neutral expression

(b) *gengive*: extrude the lips to show teeth and gums

(c) *occhiolinodx*: right eye blink

(d) *occhiolinosx*: left eye blink

(e) *cruccio*: move up the nostrils squeezing the eyebrows towards the nose

(f) *bacio*: move the lips as when giving a kiss

(g) *sorriso*: smile with teeth

(h) *sorrisino*: smile without teeth

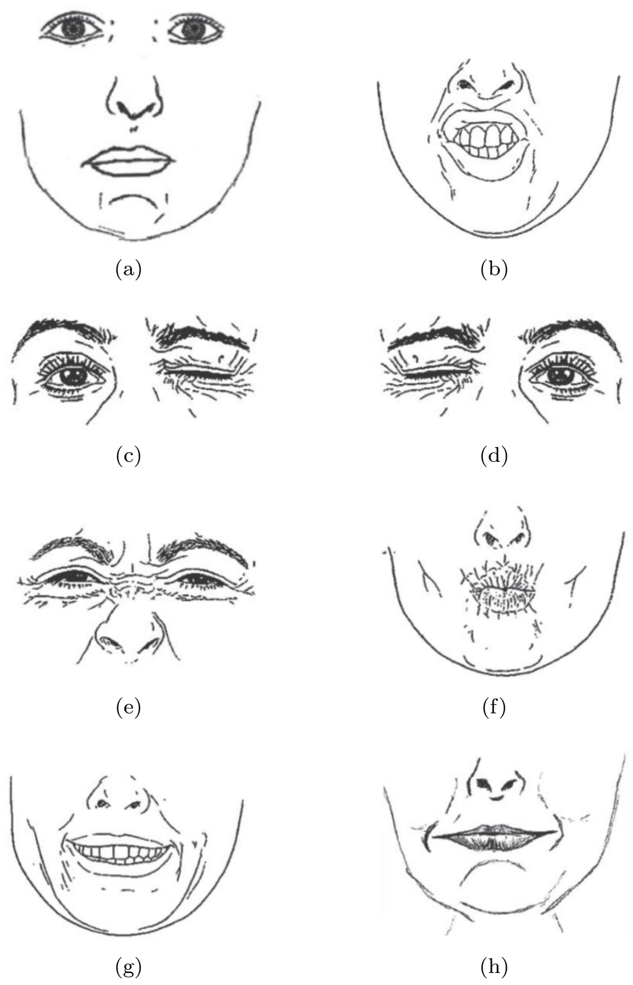


Figure 2. Dataset facial expressions

4. Proposed approaches

In this section we are going to present all the approaches we explored to measure face deformation and perform the expression recognition tasks.

4.1. Metric Learning

The first family of approaches are based on metric learning. This metric learning technique aims to automatically construct task-specific distance metric from (weakly) supervised data, in a machine learning manner. The learned distance metric can then be used to perform various task (e.g., k-NN classification, clustering, information retrieval). The metric learning problems fall into two main categories depending on the type of supervision available on the training data: *supervised learning*, *weakly supervised learning*. The metric learning problem is generally formulated as an optimization problem where one seeks to find the parameters of a distance function that optimize some objective function measuring he agreement with the training data. All metric learning algorithm we use learn so-called Mahalanobis distances. Given a real-valued parameter matrix L of shape $(num\_dims, num\_features)$, the Mahalanobis distance associated with L is defined as:

$$D(x, x') = \sqrt{(Lx - Lx')^T(Lx - Lx')}$$

In other words, a Mahalanobis distance is a Euclidean distance after a linear transformation of the feature space defined by L (taking L to be the identity matrix recovers the standard Euclidean distance). Mahalanobis distance metric learning can thus be seen as learning a new embedding space of dimensions num_dims [4].

Once learned, the metric is used for calculating the distance between the actual image taken from the webcam and the facial expression we chose. As reference landmarks for each facial expression we use the average of all landmarks of such expression in our dataset.

4.1.1 Supervised metric learning

In supervised metric learning the algorithm has access to a set of data points each of them belonging to a class (label) as in a standard classification problem. Broadly speaking, the goal in this setting is to learn a distance metric that puts points with the same label close together while pushing away points with different labels. In our case the features are the normalized landmarks and the label is the corresponding facial expression.

The supervised metric learning algorithm we tested are the following:

Large **M**argin **N**earest **M**etric **L**earning (**LMNN**) algorithm learns a Mahalanobis distance metric in the kNN classification setting. The learned metric attempts to keep close k-nearest neighbors from the same class, while keeping examples from different classes separated by a large margin. This algorithm makes no assumptions about the distribution of the data.

Neighborhood **C**omponents **A**nalysis (**NCA**) is a distance metric learning algorithm which aims to improve the accuracy of nearest neighbors classification compared to standard Euclidean distance. The algorithm directly maximizes a stochastic variant of the leave-one-out k-nearest neighbors (kNN) score on the training set. It can also learn a low-dimensional linear transformation of data that can be used for data visualization and fast classification.

Local **F**ischer **D**iscriminant **A**nalysis (**LFDA**) is a linear supervised dimensionality reduction method which effectively combines the ideas of *Linear Discriminant Analysis* and *Locality-Preserving Projection.* It is particularly useful when dealing with multi-modality, where one or more classes consist of separate clusters in input space. The core optimization problem of LFDA is solved as a generalized eigenvalue problem.

More details about all the cited supervised metric learning algorithm are available at [5].

4.1.2 Weakly supervised metric learning

In weakly supervised metric learning the algorithm has access to a set of data points with supervision only at the tuple level (typically pairs, triplets, or quadruplets of data points). A classic example of such weaker supervision is a set of positive and negative pairs: in this case, the goal is to learn a distance metric that puts positive pairs close together and negative pairs far away.

In our case we have used algorithms that learn on pairs of data points: in such algorithms we provide to each pair a binary label of $+1$ or -1. The value of this label indicates whether the given pairs is made up of similar points ($+1$) or dissimilar points (-1).

From the basis dataset we have created an augmented dataset by linearly interpolating the neutral landmarks configuration with every expression configuration for each subject. We then associated to this interpolated configuration its interpolation rate (where a higher rate gives more importance to the expression and a lower rate gives more importance to the neutral face); these interpolation rates are in the range $[0, 1]$. To avoid an excessive number of pairs we have used only the interpolation rate 0.5.
We train our weakly-supervised model considering:

- **similar pairs**: pairs with same facial expressions, same interpolation rate and different subjects

- **dissimilar pairs**: pairs with different facial expressions, same interpolation rate and also here different subjects

We can immediately notice that we employ a tiny subset of the space of all possible pairs: because of the quadratic growth of this space with respect to initial data we focused only on the pairs that we found most meaningful from a classification point of view.

The weakly-supervised metric learning algorithm we tested are the following:

Information **T**heoretic **M**etric **L**earning (**ITML**) algorithm minimizes the (differential) relative entropy, aka Kullback-Leibler divergence, between two multivariate Gaussians subject to constraints on the associated Mahalanobis distance, which can be formulated into a Bregman optimization problem by minimizing the LogDet divergence subject to linear constraints. This algorithm can handle a wide variety of constraints and can optionally incorporate a prior on the distance function. Unlike some other methods, ITML does not rely on an eigenvalue computation or semi-definite programming.

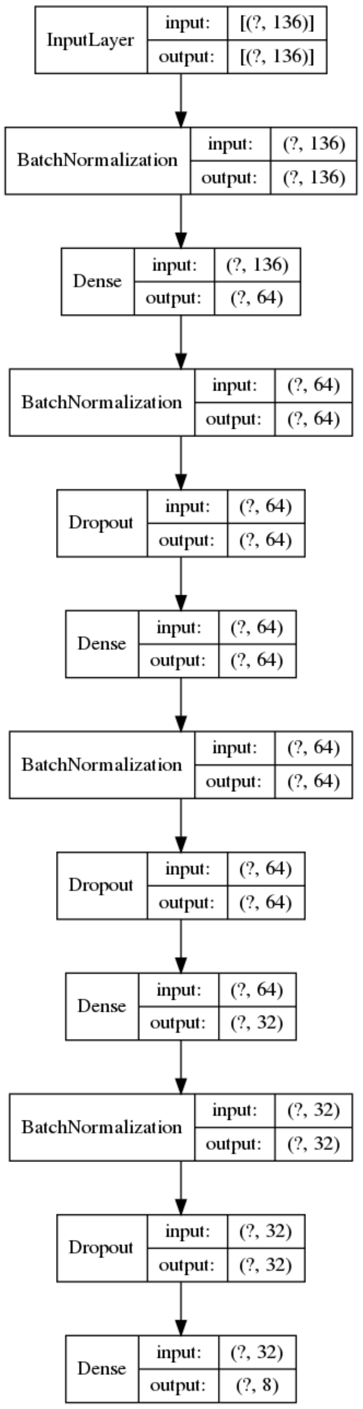
Mahalanobis **M**etric for **C**lustering (**MMC**) algorithm minimizes the sum of squared distances between similar points, while enforcing the sum of distances between dissimilar ones to be greater than one. This leads to a convex and, thus, local-minima-free optimization problem that can be solved efficiently. However, the algorithm involves the computation of eigenvalues, which is the main speed-bottleneck. Since it has initially been designed for clustering applications, one of the implicit assumptions of MMC is that all classes form a compact set, i.e., follow a unimodal distribution, which restricts the possible use-cases of this method. However, it is one of the earliest and a still often cited technique. The algorithm aims at minimizing the sum of distances between all the similar points, while constrains the sum of distances between dissimilar points.

More details about all the cited weakly supervised metric learning algorithm are available at [5].

4.2. Neural Networks

Since metric learning approaches gave interesting but not entirely satisfying results we performed a radical change of methodology. We tried to leverage the expressiveness power of neural networks in order to improve the results obtained.

Figure 3 outlines the model we used in our training. In the inner dense layer we have used the *exponential linear unit* (elu) as the activation function. In the output layer we have used the sigmoid activation function instead, which ensures that the output lies in the unit interval. The optimizer employed is the adaptive stochastic gradient descent *Adam optimizer* with learning rate equal to 1e-3. Such model leverage the high level library *Keras* using as backend *TensorFlow* [2] [3].



InputLayer | input: | [(?, 136)]
output: | [(?, 136)]
BatchNormalization | input: | (?, 136)
output: | (?, 136)
Dense | input: | (?, 136)
output: | (?, 64)
BatchNormalization | input: | (?, 64)
output: | (?, 64)
Dropout | input: | (?, 64)
output: | (?, 64)
Dense | input: | (?, 64)
output: | (?, 64)
BatchNormalization | input: | (?, 64)
output: | (?, 64)
Dropout | input: | (?, 64)
output: | (?, 64)
Dense | input: | (?, 64)
output: | (?, 32)
BatchNormalization | input: | (?, 32)
output: | (?, 32)
Dropout | input: | (?, 32)
output: | (?, 32)
Dense | input: | (?, 32)
output: | (?, 8)

Figure 3. Neural Network model

Because neural networks need a lot of data in order to work properly and given that our dataset is relatively small, during training of the net we perform a strong data augmentation which ensures that the network doesn't see the same image twice.

Such data augmentation is implemented by randomly flipping, rotating and cropping the source images at each step, moreover we also randomly chose an interpolation rate (between zero and one) which has repercussions on the labels we assign to each sample. To each sample we assign eight labels (one for each facial expression): six of them are always zero (the ones corresponding to other facial expressions), while the actual facial expression labels (the neutral and the expression) are equal respectively to the interpolation rate r and to $1 - r$. It's important to note that, in order to aid the network in learning to successfully discriminate expressions, in one every fourth sample the interpolation rate is coerced to be one.

When using the trained neural network distance computation between the image taken from the webcam and the reference landmarks of the action chosen is not needed. In fact thanks to the multilabel prediction capabilities of the network all the information needed to recognize expressions is learned during the training process. We then interpret the distance from a facial expression as $1 - p$, where p is the network output probability of such facial expression occurring (given from the neural network).

4.3. Distance on a Riemannian Manifold

Finally we try an approach which doesn't need any learning process, but simply performs a geometric transformation on the landmarks. After such transformation, which maps our landmarks on a Riemannian manifold, we compute the geodesic distance on such manifold. Similarly to the metric learning models we use such metric to compute the distance between the actual image taken from the webcam and the facial expressions chosen. Like in the metric learning models, for each facial expression we take the average of all landmarks of such expression in our dataset and use the result as the reference landmarks for that expression.

5. Conclusion and future work

We proposed multiple approaches for measuring face deformation and perform an expression recognition task: these approaches are based on supervised and weakly supervised metric learning, neural networks and geodesic distance on a Riemannian Manifold. The proposed techniques have all been evaluated and we can state that the model which gives the best results is the neural network. It can recognize almost all facial expressions – although it has some difficulties on the *blink* expression due to the limited deviation of the landmarks from the neutral position. Among the metric learning approaches the best model is by far the weakly supervised **ITML**, the other metric learning approaches output very noisy predictions and the variations of the distances measured are not very appreciable. Unfortunately the third approach (the distance calculation on a Riemannian manifold) requires more computational power in computing the geometric transformation than the one available at our disposal. Due to this fact we can not express any definitive judgment on such approach.

Possible future development may include an extension of facial expressions detected and measured, including a larger dataset which would help more so in the training of the neural network.

5.1. Resources

Code, instructions about the developed application, trained models and further details are available at `https://gitlab.com/reddeadrecovery/facestretch`.

5.2. Special Thanks

Special Thanks to all our friends who have cooperated in the creation of the dataset.

Figure 4. Dataset collage

References

[1] Curse of dimensionality. `https://en.wikipedia.org/wiki/Curse_of_dimensionality`.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin,

S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] F. Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[4] C. CJ, T. Yuan, V. William, de, B. Aurélien, and V. Nathalie. Metric learning introduction. `http://contrib.scikit-learn.org/metric-learn/introduction.html`.

[5] C. CJ, T. Yuan, V. William, de, B. Aurélien, and V. Nathalie. Supervised metric learning. `http://contrib.scikit-learn.org/metric-learn/supervised.html`.

[6] C. CJ, T. Yuan, V. William, de, B. Aurélien, and V. Nathalie. Weakly supervised metric learning. `http://contrib.scikit-learn.org/metric-learn/weakly_supervised.html`.

[7] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014.

[8] A. Rosebrock. Facial landmarks with dlib, opencv, and python. `https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/`.