

DeepFace: Closing the Gap to Human-Level Performance in Face Verification

• [[📄 taigman_cvpr14.pdf]]

• Basic Idea

- 🟡 **P1** In modern face recognition, the conventional pipeline consists of four stages: detect \Rightarrow align \Rightarrow represent \Rightarrow classify.
- This paper tries to better align the images of faces using 3D face modelling in order to apply piece-wise affine transformation.
- Also, this paper tries to learn better representations for the aligned face images.

• Face Alignment

- Revisit this portion after reading the camera and stuff.

• Representation

- Train a multi-class face recognition task (classify each face image).
- **Tricks of the trade:**
 - *The classic problem of retaining the dense spatial information throughout the network which often gets lost in the classification networks due to max-pooling operation.*
Solution: 🟡 **P3** Max-pooling layers make the output of convolution networks more robust to local translations. When applied to aligned facial images, they make the network more robust to small registration errors. However, several levels of pooling would cause the network to lose information about the precise position of detailed facial structure and micro-textures. Hence, we apply max-pooling only to the first convolutional layer.
 - *Locally connected Convolutional Features to learn different discriminative features for different spatial location.*
Solution: 🟡 **P4** The subsequent layers (L4, L5 and L6) are instead locally connected [13, 16], like a convolutional layer they apply a filter bank, but every location in the feature map learns a different set of filters. Since different regions of an aligned image have different local statistics, the spatial stationarity assumption of convolution cannot hold. For example, areas between the eyes and the eyebrows exhibit very different appearance and have much higher discrimination ability compared to areas between the nose and the mouth. In other words, we customize the architecture of the DNN by leveraging the fact that our input images are aligned.
NOTE: Increasing the number of filters will do the same job because each filter will kind of learn different discriminative feature and will fire up at different locations. The only downside of this approach is that each filter will be ran over the entire image (and hence compute intensive) whereas locally connected convolutional filter will be ran over a local patch in the image (less compute intensive).
 - *Making the representations learnt invariant to illumination changes*
Solution: 🟡 **P4** As a final stage we normalize the features to be between zero and one in order to reduce the sensitivity to illumination changes: Each component of the feature vector is divided by its largest value across the training set. This is then followed by L2-normalization: $f(I) := \bar{G}(I) / \|\bar{G}(I)\|_2$ where $\bar{G}(I)_i = G(I)_i / \max(G_i, \epsilon)$, $\epsilon = 0.05$ in order to avoid division by a small number.
🟡 **P4** Since we employ ReLU activations, our system is not invariant to re-scaling of the image intensities. Without bi- 🟡 **P5** ases in the DNN, perfect equivariance would have been achieved.
This is because it can happen that if we scale the intensities, it may go the negative

region of ReLU (which was originally in the positive region) and vice-versa. Ex. Image intensity at a location is 127, weight and bias are 1 and -126 respectively. $WI + b = 127 * 1 - 126 = 1 > 0$. Now, if we scale image intensity to 64, $WI + b = 64 * 1 - 126 = -62 < 0$. The value of ReLU will be different for both of them. Now, if $b=0$, then αWx and Wx will always be of the same sign.

NOTE: Modern networks alleviate this problem by normalizing the input by dividing by the mean and standard deviation across the whole dataset (can also do sample-wise normalization to make it illumination invariant).

• Properties of the representations learnt:

1

- It contains non-negative values (Non-negative as in 0 because they apply ReLU).
- It is very sparse. 🟡 **P4** One interesting property of the features produced by this network is that they are very sparse. On average, 75% of the feature components in the topmost layers are exactly zero. This is mainly due to the use of the ReLU [10] activation function: $\max(0, x)$. This soft-thresholding non-linearity is applied after every convolution, locally connected and fully connected layer (except the last one), making the whole cascade produce highly non-linear and sparse features. Sparsity is also encouraged by the use of a regularization method called dropout [19] which sets random feature components to 0 during training.
- Its value are between [0, 1]

• Metric Learning

- **Inner Product:** Taking the inner product between the two normalized feature representations (two query images).
- **Weighted χ^2 distance:** Because of the properties mentioned in **Properties of the representations learnt**: they use:

$$\chi^2(f_1, f_2) = \sum_i w_i \frac{(f_1[i] - f_2[i])^2}{f_1[i] + f_2[i]}$$

where f_1 and f_2 are DeepFace representations. The weight parameters are learned using a linear SVM, applied to vectors of the $(f_1[i] - f_2[i])^2 / f_1[i] + f_2[i]$.



- **Siamese Network:** End-to-end metric learning system to identify whether the features belong to the same person. 🟡 **P5** This is accomplished by: a) taking the absolute difference between the features, followed by b) a top fully connected layer that maps into a single logistic unit (same/not same). The network has roughly the same number of parameters as the original one, since much of it is shared between the two replicas, but requires twice the computation. Notice that in order to prevent overfitting on the face verification task, we enable training for only the two topmost layers. The Siamese network's induced distance is:

$$d(f_1, f_2) = \sum_i \alpha_i |f_1[i] - f_2[i]|$$

where α_i are the trainable parameters trained using standard cross-entropy loss and back-propagation of the error. Note that this induced distance is because they are taking the difference between the feature representations.

• Network with Best Results

- Ensemble of 3 networks trained with different inputs:
 - 3D aligned RGB images
 - The gray-level images plus image gradient magnitude and orientation
 - 2D aligned RGB images

-  **P7** We combine those distances using a non-linear SVM (with $C=1$) with a simple sum of power CPD-kernels: $K_{\text{Combined}} := K_{\text{single}} + K_{\text{gradient}} + K_{\text{align2d}}$, where $K(x, y) := -\|x - y\|_2^2$, and following the restricted protocol, achieve an accuracy 97.15%. 

► Unlinked References

