ACTORS RECOGNITION AND RETRIEVAL IN MOVIE SCENES

Bianco Michael, Bertolini Matteo, Todaro Marco
{270801, 312516, 270643}@studenti.unimore.it
University of Modena and Reggio Emilia

Abstract

We present an approach to detect and recognize automatically the actors present in a movie scene. This method is in general applicable to all systems where the task is to identify a person between a list of other subjects, the only need is to have a proper dataset and fine-tune the model on these new data. The algorithm also makes use of a generator able to reconstruct the frontal face if an actor is standing sideways or if is wearing something that covers his face, for example a cap or a pair of glasses. Similar task is performed by Amazon Prime video X-Ray but we focus on the single frame. This project can help new video services to provide better user experiences, for instance displaying other useful information on the actor detected or similar movies in which he/her acts.

1. Introduction

We built a deep learning algorithm to recognize and detect celebrities acting in different movie scenes, even if they are dressed up, made up or in movement. The literature necessary to work on this kind of model requires the study of convolutional neural networks for extracting the most salient features useful to describe each actor we are recognizing and algorithms for face detection.

1.1. The problem

Have you ever watched a movie and didn't recognize an actor, so you searched the internet for their name but you didn't find it?
We worked to provide an integrated, AI-based solution with names of all actors for each movie frame, able to find a specific actor between a list of 544 persons and over 10000 images.

2. Proposed methods

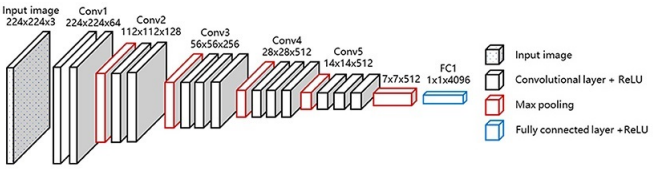
Starting from the original frame, YOLOFace [1] detects the faces in the scene. Each of them is preprocessed, reshaped and then passed through the network, which is a VGGFace model with only the first fully connected layer. The model provides an embedding in a latent space of 4096 dimensions, which we project in another space with Principal Component Analysis (PCA) [2] of 1024 dimensions to improve the speed of the eleboration. The inference is done by retrieval of the most similar items in the dataset, and weighted max voting is performed on them to classify the correct actor name. In case the actor is standing in profile, the image is passed through a generative network able to estimate the frontal view of the face to improve the classification.

2.1. Dataset

Starting from the FaceScrub [3] dataset, we manually added others missing actors, reaching a total of 544 actors. For each of them we used 22 images of size 224x224 RGB, 20 for the train set and 2 for the test set, for a total of 10880 images.

2.2. Model

The model is a VGGFace CNN truncated to the first fully connected layer made of 4096 neurons in output (which is actually the same as VGG-16 truncated to the first fc layer). The number of learnable parameters is 117.544.896.



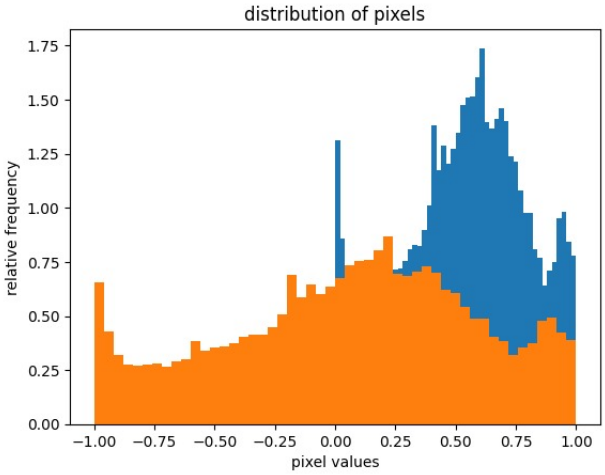
2.3. Training

The model was pre-trained using the VGGFace model on LFW dataset [4].
We performed fine-tuning for 41 epochs on the first fully connected (fc) layer. In an attempt to further decrease the loss, we additionally conducted 15 epochs of fine-tuning on

all the previous layers and saw an increase of the performances. We trained on 3 GPUs in parallel with a batch size of 16 elements, so for each epoch, the 544x20 = 10880 images are divided into 226 steps on each GPU.
Every image is preprocessed with the following transformations:

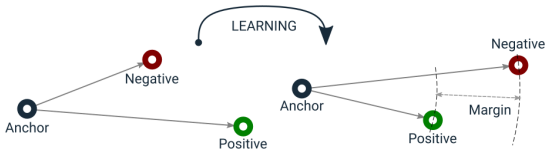
- RandomHorizontalFlip()
- RandomGrayscale(p=0.2)
- Normalize((0.5,), (0.5,))

The first horizontally flips the given image randomly with a given probability, taking to account the case in which the actor is standing with the face in the opposite direction with respect to the one in the training image. The second randomly converts the image to grayscale with a probability of p, making the network robust even on luminance variations or black and white scenes.
Then at the end we normalize along each channel with : image = (image - mean) / std. The parameters mean and standard deviation are passed as 0.5, 0.5 to normalize the image in the range [-1,1]. For example the minimum value 0 will be converted to (0-0.5)/0.5=-1, the maximum value 1 will be converted to (1-0.5)/0.5=1, in this manner we normalize the distibution of the pixels and increase the value range to perform a better classification.



The Loss function of the model is the triplet loss, its goal is to learn a representation space where similar examples are closer together and dissimilar examples are farther apart.
In triplet loss, each training example consists of an anchor, a positive example, and a negative example. The anchor is an example from the dataset, the positive example is an example similar to the anchor (e.g., the same person's face), and the negative example is a dissimilar example (e.g., a different person's face). The triplet loss function computes the distance between the anchor and the positive example and compares it to the distance between the anchor and the negative example. The model is then trained to minimize the distance between the anchor and the positive example while maximizing the distance between the anchor and the negative example. In this way we obtain an embedding space where similar examples are closer together, so if a new image is projected near the images of the same actor, we can take the nearest most voted images to identify the person.

The loss function for each sample in the mini-batch is:
$L(a, p, n) = max\{d(a_i, p_i) - d(a_i, n_i) + margin, 0\}$



2.4. Evaluating results

As described before we divided the dataset in 20 images per actor for training and 2 for testing.

From the graph we can see that the training-loss and the testing-loss decrease a lot in the last 15 epochs in which we backpropagate the gradient through all the network. The minimun loss on test-set is reached on epoch 47.
These weights are used to perform inference on new examples.
After that we looked at the embeddings generated by the model and plotted some examples on 10 actors to validate the work done and ensure that different examples are divided into well-distinct clusters.
To plot the 1024-dimensional space into a 2d space we used T-sne method, here are some results.

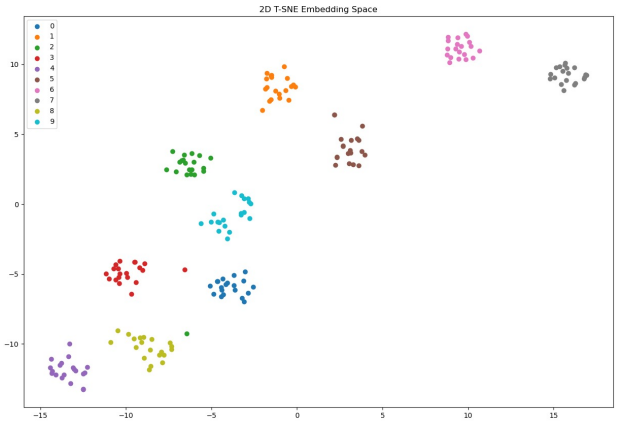


Figure 1. T-sne 1024 embedding space on 10 actors

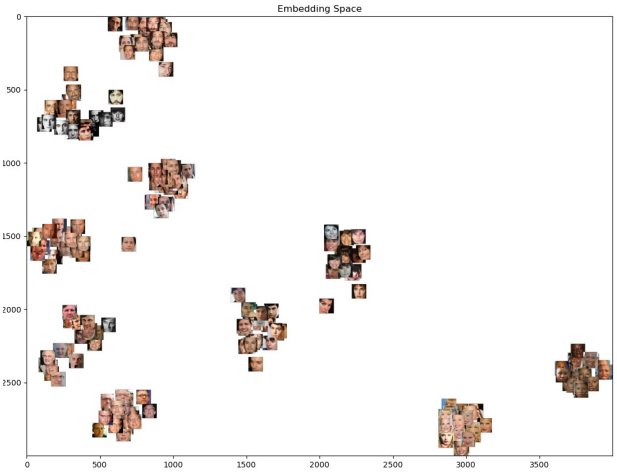


Figure 2. T-sne 1024 embedding space on 10 actors with training images

2.5. Inference

Starting from an HD video, for the single frame, we resize it by half in both dimensions. This because YOLO performs better with small frames given its small bounding boxes. The inference pipeline starts by extracting a single frame from a movie scene and passing it through YOLO-Face network to perform face detection.
YOLOFace is a face detection algorithm based on the YOLO architecture, it uses a deep CNN to predict bounding boxes and class probabilities for faces. It divides the input image into a grid, predicts multiple bounding boxes and class probabilities per grid cell, applies non-maximum suppression, and outputs the final face detections with associated attributes or categories.

Given a bounding box we project it back to the original HD frame, to obtain a bounding box grater than 224x224 pixels. In fact we noticed that the following reshape to 224x224 from a smaller bounding box stretches the image and compromises the final classification.
Then we apply the same normalization as in the training step to change the histogram distribution with transforms.Normalize((0.5,), (0.5,)).
At this point the tensor is passed through the network to obtain a feature vector of 4096 dimensions, which is reduced by the PCA to 1024 dimensions.
The inference is provided by a custom Nearest Neighbors algorithm made by us, which performs a weighted 5 max-voting measure and returns the index of the predicted actor and a list of the nearest 5 neighbors with their class.

2.6. Nearest Neighbors Hyperparameters

We tried different distance measures to obtain the best result on the test-set.

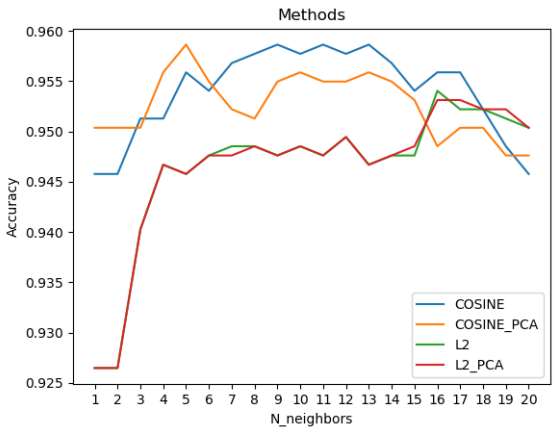


Figure 3. Test-set accuracy with L2 and Cosine distance

In particular, we tried both the L2 distance and the cosine distance with and without PCA, and found that the best accuracy is given by cosine distance + PCA with 5 neighbors, accuracy: 95,86% on 1088 images in the test set.

$$\text{cosine distance} = 1 - \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| * \|\mathbf{B}\|}$$

3. Elements of geometry with GAN

In most of the situations the images acquired from the video are not frontal at all but instead are sideways. For example imagine when we watch a video and we see only one side of the face of an actor talking with another sideways too. We cannot capture this face frames to calculate the inference and recognise the actor because the classifier would

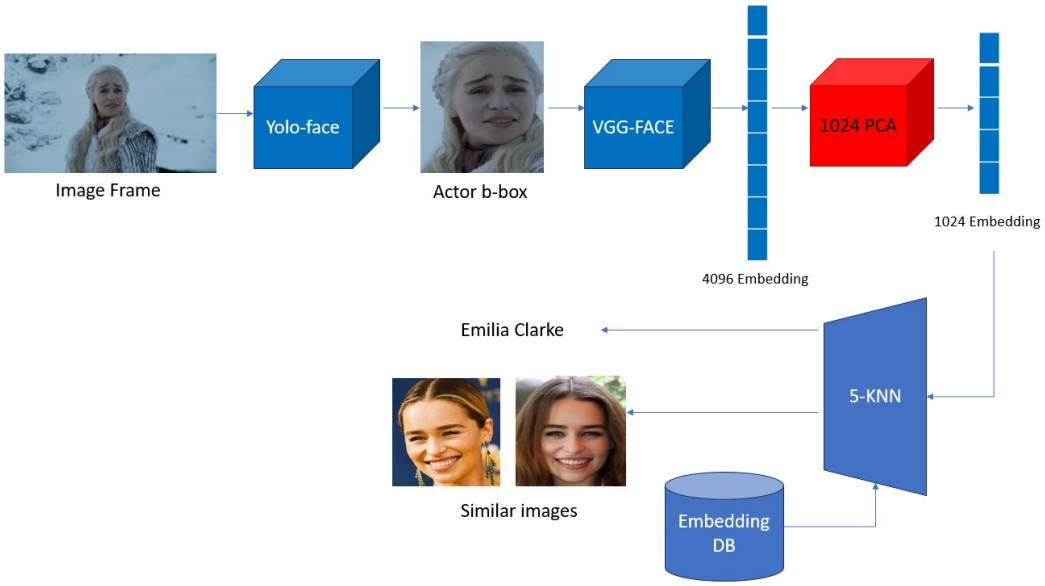


Figure 4. The Inference Pipeline

not work with high accuracy. This because the train set of the images we are using to compute the model is all made of frontal and clear images of the subject. Furthermore the frame captured can present elements that cover the actor's face like cap, sunglasses, scarf and so on... For this reason we implemented in our DL architecture a pre-trained CFR-GAN [5] that works to solve the problems above. In general the GAN performs two tasks:

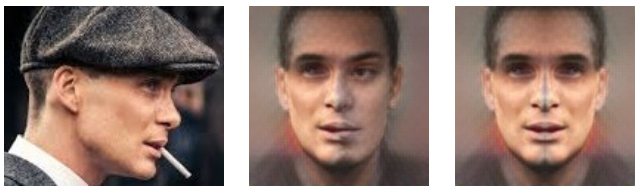
- Align the detected face and reconstruct the hidden side to reconstruct the whole face;
- Remove the "obstacles" that hide some face parts.

3.1. GAN Structure

The input image is passed through a 3D reconstruction and de-occlusion model to extract shape and texture features of the face, then the points are projected inside a MATLAB 3D model to obtain two images, one letting the occlusions (ex. glasses) and one without them. The training is done in a self-supervised manner by passing the two images through a Swap-Rotate&Render step which tries to rotate and merge the two images to obtain a frontal reconstructed face, which is then passed through a generator network to create and render the initial image. Once trained, during testing, the two profile images are rotated and passed through the generator to render the frontal image of the face.

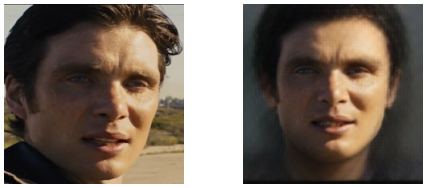
3.2. Results with CFR-GAN

We saw that if the actor is standing in profile the visible part is reconstructed very well, while the hidden part is less precise, so in these cases the visible part is mirrored. If more part of the face is visible this step is not necessary, because the image reconstructed has already a better quality. The use of the GAN is useful if the actor is standing in profile to perform a better classification, instead if we have the frontal image it could be useful to clean it from occlusions, but in general in these cases the use of the GAN is not necessary. Below there are some results.



(a) Profile image (b) GAN reconstruction (c) Image mirrored

Figure 5. Different results on reconstruction



(a) Frontal image (b) GAN reconstrucion

Figure 6. Reconstruction with frontal image

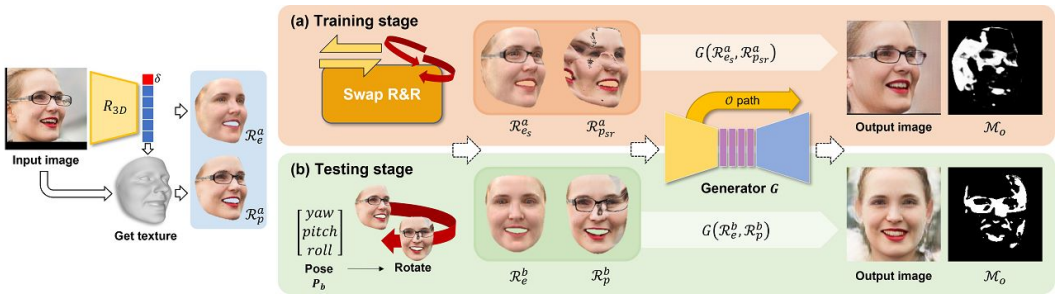


Figure 7. CFR-GAN architecture

4. Different Approaches

At the beginning of the project we started with the full VGGFace architecture where we changed the dimension of the last fully connected layer from 2622 to 28 neurons, one for each of the actors in our inital dataset created manually. The task of classification is limited to the number of possible actors in the dataset while the task of retrieval is more flexible and doesn't affect the network structure.
In fact if we need to add more actors we just need to fine-tune the same model without changing its structure, mandatory operation on the classification task.

5. Zero-Shot

We tried to add other two actors without re-training or fine-tuning the network and saw that using the same mechanism used for our dataset (projecting the images in the 4096 space, applying the PCA and the same 5 KNN with the cosine distance), the network was able to classify these two new persons. If we had used the classification task instead of the retrieval, there was no way to obtain a correct classification without changing the dimension of the last fully connected layer and fine-tuning again the network. In fact our architecture learns how to project similar images near themselves in order to obtain a similar representation of the same person and a different representation for different identities. It is possible that continuing adding other new persons it can be necessary to fine-tune again the network using the triplet loss.

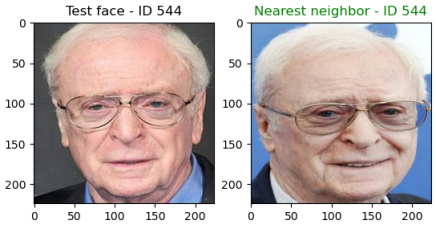


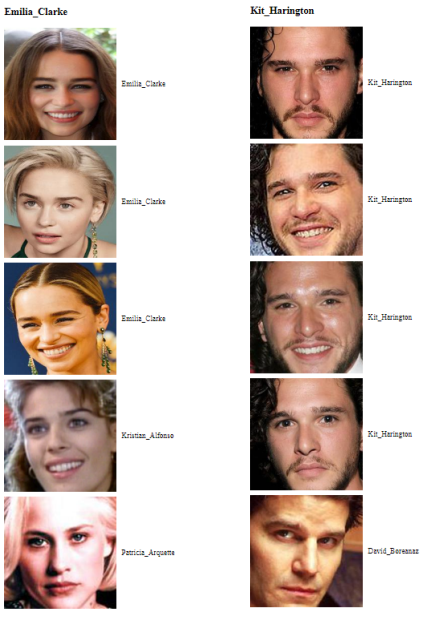
Figure 8. Result on Michael Caine (not in the train set)

6. DEMO and testing on real movies

We developed also a live demo example that allows the user to test the system with real movies extracting predictions on single frames. The demo displays:

- the detections found in the frame
- relative predictions
- five nearest images for each actor

It is possible to detect and classify multiple actors in the single frame. If the system is sure about the predictions (at least 3 same predictions) the bounding box is green, instead if there are 2 same predictions it is yellow, else it's red. The classification starts when the user pauses the video on the scene where there is at least one face detected by YOLO. If there is no detection found, a proper message is displayed.



(a) Emilia Clarke (b) Kit Harington

Figure 9. 5 Nearest Neighbors

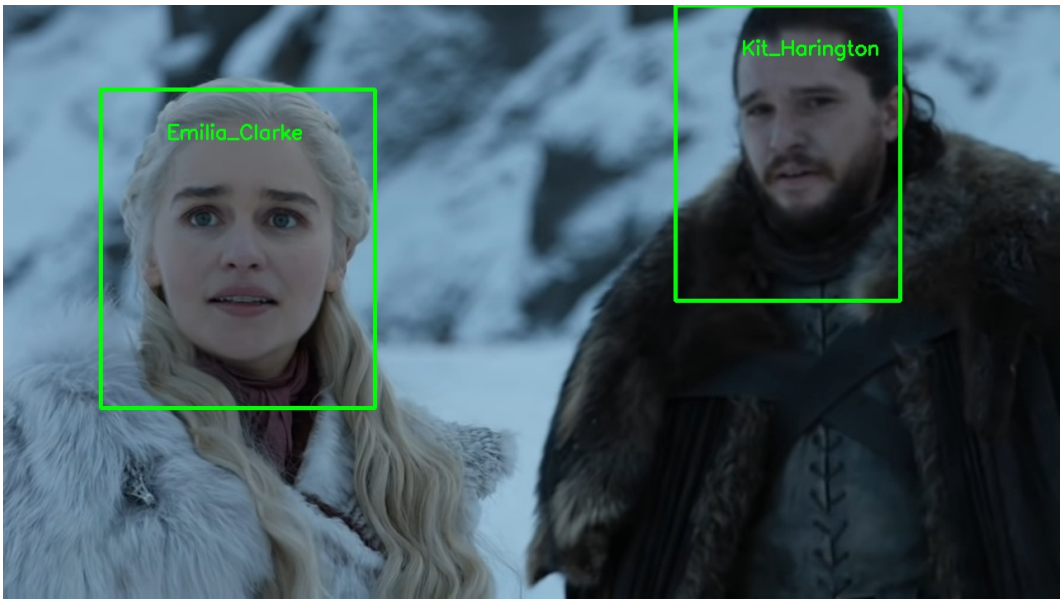


Figure 10. Multiple predictions

7. Limitations

The model isn't perfect on real movies when the frame is 720p or less because the bounding box predicted by YOLO is more or less rectangular, not centered on the actor face and a further resize to a square detection with shape less than 224x224 pixels causes a corruption of the face image, with consequent misclassification. There could be other errors in classification if the actor is moving and the image is blurred due to the movement.

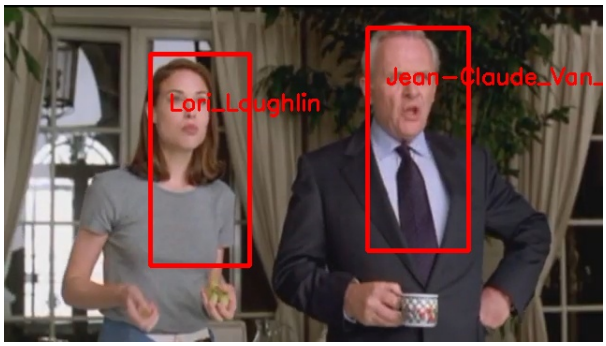


Figure 11. Misclassifications due to YOLO and small frame size

8. Conclusions

We are planning to merge the GAN into the inference pipeline in order to classify directly if the actor is standing in profile. Another step is working on the motion of the scene, taking to account the time factor to get an even better prediction and avoid misclassifications if an actor is moving and his detection is corrupted.

In the end, in our solution we reached satisfying results and we built a model that can be used on a generic people recognition task, able to identify anyone between hundreds of other persons analysing the most similar images we have.

References

[1] Weijun Chen, Hongbo Huang, Shuai Peng, Changsheng Zhou, and Cuiping Zhang. Yolo-face: a real-time face detector. *The Visual Computer*, 37:805–813, 2021. 1

[2] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. 1

[3] Hong-Wei Ng and Stefan Winkler. A data-driven approach to cleaning large face datasets. In *2014 IEEE international conference on image processing (ICIP)*, pages 343–347. IEEE, 2014. 1

[4] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015. 1

[5] Lin Pengyue, Yang Wen, Xia Siyuan, Jiang Yu, Liu Xiaoning, and Geng Guohua. Cfr-gan: A generative model for craniofacial reconstruction. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 462–469, 2021. 4