DeepFakes Detection Lab - Report

Luis Antonio Ortega Andrés
Antonio Coín Castro

March 23, 2021

Task 1: intra-database analysis

The goal of this task is to develop and evaluate DeepFake detection systems over the same database. In this task, you should use only the UADFV database, which is divided into `development` *and* `evaluation` *datasets.*

The notebook associated with this task is `Task1-2.ipynb`.

a) *Provide all details (including links or references if needed) of your proposed DeepFake detection system.*

Our DeepFake detection system consists on the following modules:

Preprocessing

First of all, the given images from UADFV have been preprocessed following these steps:

1. The data is loaded using `Keras' flow_from_directory` function (Chollet *et al.*, 2015), so that we don't have to keep all images in memory while training our model. We resize each image to a resolution of 768×768.
2. We use the MTCNN face detector (Zhang *et al.*, 2016) in order to crop the section of the image that contains the face. The aim of this phase is to erase those parts of the image that are not relevant for our classification task.
3. We add random noise to the images as a form of regularization, to force our classifier to learn more relevant features. More precisely, we randomly decide wether the image is slightly blurred using a Gaussian filter from OpenCV (Bradski, 2000), or simply perturbed by Gaussian noise. Each of these outcomes is equally likely and every image goes through one or the other, but never both. Note that these perturbations are only applied to training samples and not to the evaluation images.

Feature extraction

We decided that the best way of extracting features from the faces was `dlib`'s 68-point face landmark detector (King, 2009). For each (cropped) face image, we obtain the spatial position of the 68 landmarks, and our final feature vector is composed of the gray-scale intensities of these landmarks. We then center each feature vector to have zero mean.

We also considered the possibility of applying PCA for dimensionality reduction, but since we wanted to try both approaches, we delegated this decision to the actual training step.

Classifier

The final step of our model is a machine learning model that acts as a classifier to which the feature vectors are fed for training. At first we considered four distinct models for this job: a **Linear SVM**, a **SVM with RBF kernel**, a **2-layer perceptron** and a **Logistic Regression** model. The process by which a final classifier was selected can be seen on the next section.

b) *Provide all details of the development/training procedure followed and the results achieved using the development dataset of the UADFV database. Show the results achieved in terms of Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC).*

For the training procedure we performed a hyperparameter grid search using `Sklearn`'s API (Buitinck *et al.*, 2013), in which each family of classifiers mentioned above is tested with a wide range of possible hyperparameters to maximize the AUC metric. For example, the number of hidden layers considered for the MLP are $\{(50), (100), (50, 50), (100, 100)\}$, and the linear SVM regularization term C varies from 10^{-3} to 10^{3} with 40 equidistant steps. To make the process fair, we employ **5-fold cross validation** to select the best hyperparameters for each model, and in the end we select the model that achieved the highest cross-validated AUC. Note that each family of classifiers is trained using the same folds, so the results are comparable.

Before completing the training pipeline, we added the possibility to train each classifier **with and without PCA reduction**, to see whether it would have a positive impact on the final result. The fraction of explained variance to be retained was also an hyperparameter, whose search space was $\{0.90, 0.95, 0.99\}$. An important detail is that the data is first standardized to have zero mean and unit variance, since we observed that the performance of the classifiers was noticeably better after this normalization.

The best models of each family after running the grid search are shown in Table 1, along with their cross-validated AUC scores.

Model	**PCA**	**Parameters**	**AUC (CV)**
SVM RBF	No	$C = 1.6681,\ \gamma = 0.0167$	0.9937
SVM Linear	No	$C = 0.0174$	0.9186
MLP	No	$\alpha = 0.2154,$ size=$(100, 100)$	0.9883
LR	No	$C = 0.1624$	0.9180

Table 1: Results of the hyperparameter tuning phase.

As we can see, the best performing model was the **non-linear SVM with RBF kernel using the raw features**, and this is the model that we choose as our final classifier (with the optimized parameters found). An interesting thing to note is that PCA was discarded on every model, probably because the number of raw features considered is not very high. The results of evaluating our chosen model on the whole development set can be seen on Figure 1.

c) *Describe the final evaluation of your proposed DeepFake detection system and the results achieved using the evaluation dataset (not used for training). Show the results achieved in terms of ROC curve and AUC. Provide an explanation of your results.*

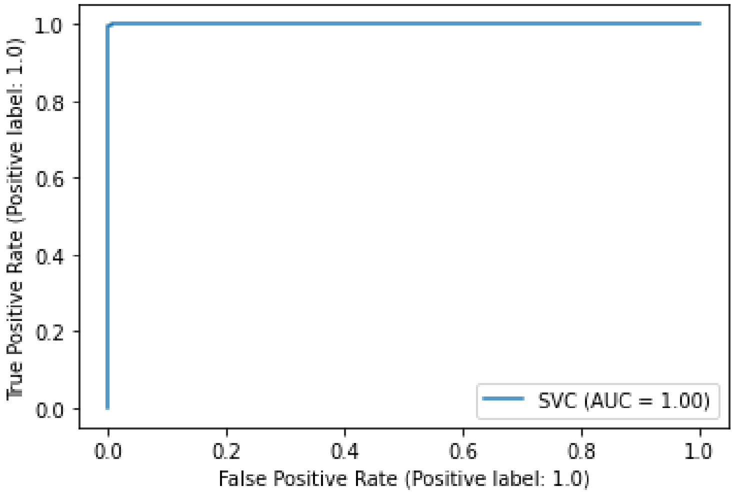

Figure 1: ROC curve and AUC value on UADFV's development set.

Having chosen our model, we simply evaluate it on the previously unseen evaluation set of UADFV database, and the results can be seen in Figure 2. We obtain an AUC of 0.95, quite close to the perfect score of 1.0. We can say that we have succeeded in this task, since our model has learned to detect DeepFakes on the UADFV database at different threshold while avoiding overfitting.

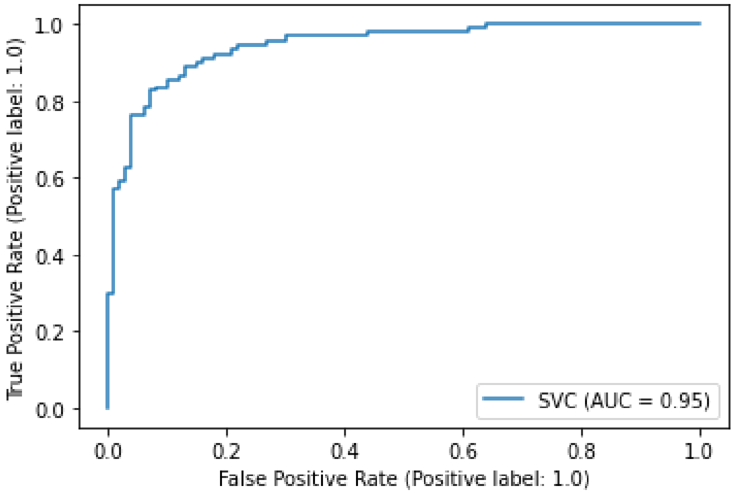

Figure 2: ROC curve and AUC value on UADFV's evaluation set.

Other approaches and failed attempts

1. We considered extracting features using an image destriptor such as SURF or SIFT, and then following the Bag of Visual Words technique. However, the results obtained were worse, so we discarded this approach.

2. Along with the landmarks intensity, we made some attempts at using the the landmarks locations as features, but this resulted in overfitting. As a regularization technique, we cen-

tered those locations (substracting the mean) in order to make them location invariant. The performance improved a bit but in the end we decided to stick to the face landmarks only.

3. At one point, MTCNN did not correctly detect the faces for every image, and during training we decided to simply skip these images. However, since we cannot skip test cases, we decided that an image with no face detected would be assumed to have the same features as the last image with the same label. In the end we solved the underlying problem (by resizing the input images), but we left all the code checks just in case they were needed in the future.

4. More models were tested in the grid search, such as Random Forests and other boosting techniques. However, they performed so poorly compared to the rest that we decided not to include them in the final report.

Task 2: inter-database analysis

The goal of this task is to evaluate the DeepFake detection system developed in Task 1 with a new database (not seen during the development/training of the detector). In this task, you should use only the Celeb-DF. You only need to evaluate your fake detector developed in Task 1 over the `evaluation` dataset of Celeb-DF, not train again with them.

The notebook associated with this task is `Task1-2.ipynb`.

a) *Describe the results achieved by your DeepFake detection system developed in Task 1 using the evaluation dataset of the Celeb-DF database. Show the results achieved in terms of ROC curve and AUC. Provide an explanation of your results in comparison with the results of Task 1.*

We evaluate the model established in the previous task (SVM with RBF kernel and optimized hyperparameters) on the evaluation set of Celeb-DF. The results achieved can be seen in Figure 3.

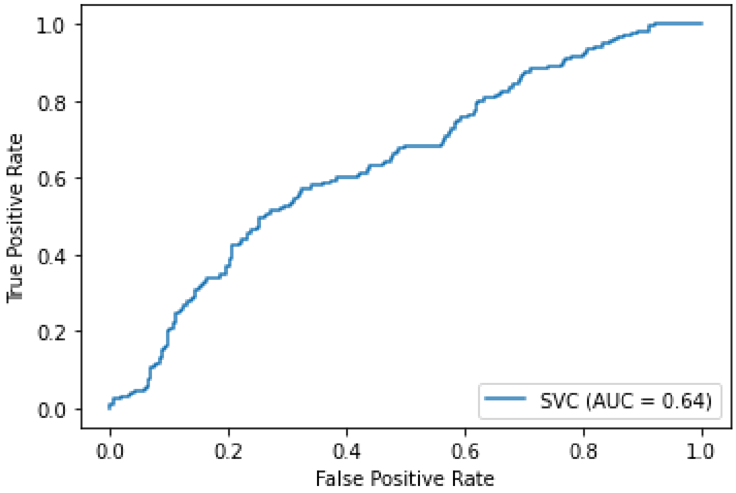

Figure 3: ROC curve and AUC value on Celeb-DF's evaluation set.

We obtain an AUC of 0.64, which is considerably lower than the one achieved in the previous task. This behaviour makes sense, considering that the training dataset was less challenging than this one, in the sense that the deepfakes were more easily spotted by a human. In the UADFV dataset there were usually vertical and horizontal strips surrounding the place where the face exchange had taken place, and in this new dataset deepfakes have a higher quality and do not show such markings. In short, our detection model scales poorly to this dataset because the data in which it was trained **is not representative enough** of this new evaluation set.

Task 3: inter-database proposal

The goal of this task is to improve the DeepFake detection system originally developed in Task 1 in order to achieve better inter-database results. You must consider the same evaluation dataset as in Task 2 (i.e. the `evaluation` *dataset of the Celeb-DF database).*

a) *Describe the improvements carried out in your proposed DeepFake detection system in comparison with Task 1.*

Train the same model with more data

`Task3_ML.ipynb`

Describe from where we have obtained the data (database, scripts to extract frames from videos, number of new examples, references).

AUC obtained: 0.67 (only the number, no graph)

This approach is not good enough.

Consider alternative features

Bib-citation to Facenet. Contar que hace face embedding a 128 características.

AUC obtained: 0.7 (only the number, no graph)

`Task3_Facenet.ipynb`

Train a deep learning model from scratch

`Task3_DL.ipynb`

Fracaso absoluto. contar modelo usado por encima.

0.1 Fine-tune a pre-trained deep learning model

`Task3_Finetune.ipynb`

contar proceso de finetuning.

3 épocas + 15 épocas.

Bib-citation to VGG19.

b) *Describe the results achieved by your enhanced DeepFake detection system over the final evaluation dataset. Show the results achieved in terms of ROC curve and AUC. Provide an explanation of your results in comparison with the results of Task 2.*

Final graph (AUC 0.81).

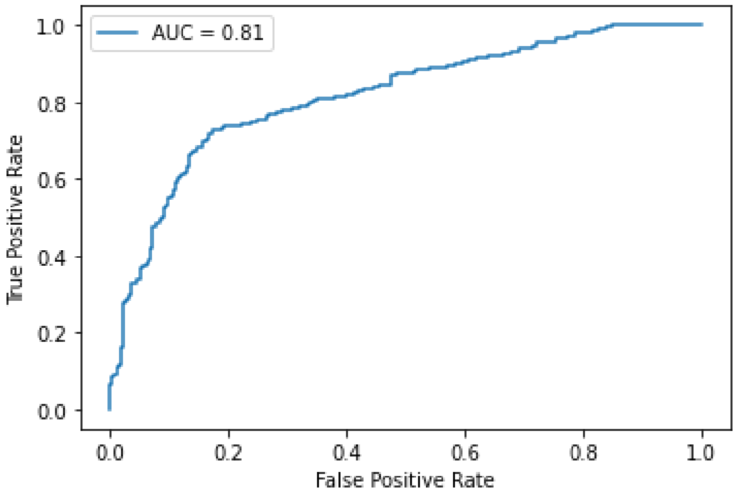

Figure 4: ROC curve and AUC value on Celeb-DF's evaluation set for our enhanced model.

c) *Indicate the conclusions and possible future improvements*.

Not enough data; finetuning is essential; Deep learning outperforms classical approaches when used appropriately.

face align, face masks, one-to-many, many-to-one, (eye, mouth, teeth,... more fine grained approach to feature extraction.)

References

Bradski, G. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Buitinck, Lars, Louppe, Gilles, Blondel, Mathieu, Pedregosa, Fabian, Mueller, Andreas, Grisel, Olivier, Niculae, Vlad, Prettenhofer, Peter, Gramfort, Alexandre, Grobler, Jaques, Layton, Robert, VanderPlas, Jake, Joly, Arnaud, Holt, Brian, & Varoquaux, Gaël. 2013. API design for machine learning software: experiences from the scikit-learn project. *Pages 108–122 of: ECML PKDD Workshop: Languages for Data Mining and Machine Learning*.

Chollet, François, *et al.* 2015. *Keras*.

King, Davis E. 2009. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, **10**, 1755–1758.

Zhang, Kaipeng, Zhang, Zhanpeng, Li, Zhifeng, & Qiao, Yu. 2016. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, **23**(10), 1499–1503.