The Deepfake Detection Challenge (DFDC)

Settimo A.
Matricola 262710
andrea.settimo@studenti.polito.it

Tabacaru R.G.
Matricola 267581
ralucagabriela.tabacaru@studenti.polito.it

Tolomei L.
Matricola 267638
leonardo.tolomei@studenti.polito.it

Abstract

Deep learning has given rise to technologies that would have been thought impossible only a handful of years ago. Manipulation of visual content techniques evolved a lot in the last few years. In particular, deep fakes became an hot topic, because they may have an huge impact on various aspects of our society. Therefore, research is now focusing on detection of such manipulated visual content.

We used XceptionNet to classify the dataset provided by the Deepfake Detection Challenge (DFDC). Our intent was to improve the accuracy of the prediction of deep-fakes.

1. Introduction

Now a days the quick evolution of the technology has led to new techniques for manipulating images and videos. Through these new approaches it is possible to modify and exchange people's faces producing deepfakes. Compared to the manipulation of images of last 10 years the innovation is the realism of the images obtained with the new techniques of artificial intelligence and machine learning.

Deepfakes use a generative adversarial network (GAN), which trains a generator and a discriminator in an adversarial game. The generator crates new images from the source material. The discriminator attempts to determine if image is generated or not. In this way the generator starts to create new images equal to the source by relying to the differences found by the discriminator. This makes deepfakes difficult to detect because any time a defect is determined, it can be corrected.

Furthermore there is an application called FakeApp, which allows anyone to create deepfakes and to share these creations online. This has resulted in a large amount of deepfake videos and images. Politics and famous people are mostly affected by this. The most famous example is the fake video in which Barack Obama [1], former American president, says words that belong to another audio track. This phenomenon lead to doubt of the truthfulness of the news, but lead more to deceive people. Because the fake new go viral on social media and on television compared to real news.

For these reasons, research, in computer vision, has moved on to recognize deepfakes. The Deepfake Detection Challenge goal is to incentive the rapidity of innovation in this area, by inviting the participants to create new ways to detect manipulated media.

2. Previous work and proposed improvements

We are going to reproduce the structure of FaceForesics++ classification [2]. Their work consisted in two parts: the first was face detection and the latter emploied pre-trained XceptionNet. In FaceForensics++ classification, XceptionNet was pre-trained on ImageNet and trained on their dataset. The first improvement that we present in our research is transfer learning with FaceForensics++ pre-trained XceptionNet and the new dataset given by the competition. We also tried to apply different techniques, such as data augmentation and pre-processing transformations, evaluating our results against our validation set and testing against the Public Test Set of the challange. Then, we tried transfer learning also starting from ImageNet weights, to make a comparison between the two methods.

3. Dataset

For the challenge was build a special dataset [3], the actors of the videos have been selected with different characteristics as skin tone and age. Each of them submitted a set of videos, which present different lighting conditions, head poses and backgrounds.

These videos have been manipulated using a number of state-of-the-art methods. A part of face swaps were performed between subjects with similar characteristics.

From each video, 15 second clips have been extracted. Only for the test set, appropriate augmentations were applied on two-thirds of the clips. The augmentations applied are: reduce the FPS of the video to 15, reduce the resolution of the video to 1/4 of its original size and reduce the overall encoding quality.

3.1. Dataset of the challenge

The site of the challenge is Kaggle in which there is a description of the dataset of the competition [4]. It is divided into: Training Set, Public Validation Set, Public Test Set, Private Test Set.

The Training Set is available for download in order to build the models, and it is just over 470 GB. The Public Validation Set is a small set of 400 videos, it is used when you commit the Kaggle notebook. The Public Test Set is completely hidden and it is used to calculate the public leaderboard. The Private Test Set is privately held outside of Kaggle's platform and it is used to calculate the final private leaderboard. It contains videos with a similar format as the other datasets.

3.2. Face extraction

We performed the extraction of faces from Training Set using the dlib library to compose the dataset on which we did the analysis. For each video, a frame every 10 seconds was extracted. The same approach of FaceForensics [5] was used, with the difference that they pulled out one frame every second.

Our decision to extract a frame every 10 seconds was influenced by the size of the dataset that affects the training time. In this way we drastically reduced the duration of training. Given the large amount of video to be processed, we had performance problems, since dlib uses CPU to extract faces. This resulted in 6 days of execution, despite having parallelized the computation. In Figure 1 we reported some of extracted fake faces with their originals.

Figure 1: Some extracted faces. Upper ones are real. The others are fake.

3.3. Data structure

To have more generalization, a particular data structure is used during training. The data structure consists of grouping the frames of the dataset by their original video. When you want to access an element of the dataset you randomly take a frame present in the frame vector of the video. The result is implicitly a data augmentation, which allows to always see different frames for each video at each time.

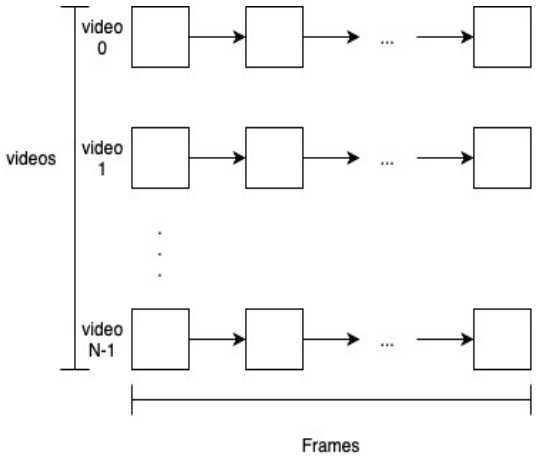

Figure 2: Frame per video

4. Metrics

4.1. Accuracy and F1-score

Two different metrics were used to evaluate the model during training: accuracy and F1-score.

The F1-score takes into account the importance of false positives and false negatives since the formula is a combination of precision and recall. In this case, the precision used is weighed on the number of classes to take into account that in the data set the two Real and Fake classes are unbalanced.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$P = \frac{TP}{TP + \frac{x}{y} \cdot FP}$$

Where:

x: is the size of Real class;

y: is the size of Fake class;

$$R = \frac{TP}{TP + FN}$$

$$F1_{score} = 2 \cdot \frac{P \cdot R}{P + R}$$

To select the best model based on these two metrics, we can combine them together by averaging them because we want to obtain a model that takes into account the importance of false positives and is also accurate at the same time.

4.2. Cross Entropy Loss

Our problem is based on binary classification, this entails choosing the Cross Entropy Loss function as the function to be minimized. The goal is to reduce the function until it reaches 0. Below, it is shown the formula:

$$CELoss(y, p) = -(ylog(p) + (1 - y)log(1 - p))$$

The loss computed by Kaggle on our submissions had an equivalent definition.

5. XceptionNet

Our case study is based on the use of the XceptionNet [6], created by Google. The goal is to use the weights in the GitHub repository of FaceForenics++ as a starting point and adapt it to use transfer learning. After that we will be able to recognize if a video was manipulated or not.

Also, to see the comparison with the performance generated with the FaceForensics++ weights, we included a transfer learning with a pre-trained network on ImageNet (unfortunately not completed due to time reasons).

6. Transfer learning on FaceForensics++ weights

6.1. Coarse-to-fine Cross Validation

In FaceForensics++ repository there are different types of weights of the XceptionNet:

Raw: weights obtained using raw images;

c23: weights obtained using videos with medium compression;

c40: weights obtained using videos with high compression.

The c23 are the weights chosen by us as a starting point.

To search for hyperparameters, we opted to use a coarse-to-fine cross-validation with the use of a random pattern. The choice of this type of search is given by the fact that this scheme allows you to avoid losing values that are not considered by the grid search methodology (Figure 3).

	Adam	RMSprop	SGD + momentum
Lr	0.0003	0.0002	0.0006
WD	0.00015	0.00004	0.00028
Accuracy	0.8080	0.8148	0.5584
F1-score	0.6067	0.6148	0.5410

Table 1: Best results of coarse phase

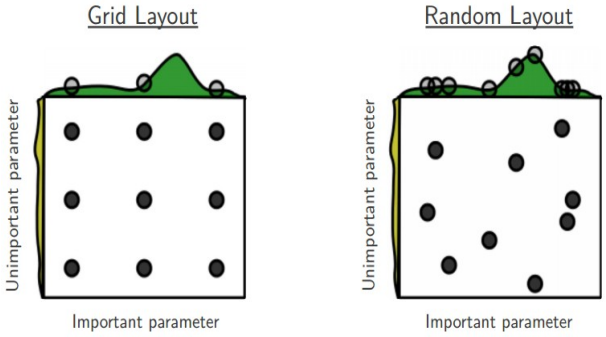

Figure 3: The search for hyper parameters

For both searches a maximum number of 50 iterations was used and going over 5 epochs on the number of data equal to:

Training set: given by 5000 videos (2500 real and 2500 fake);

Validation set: given by 5000 videos (2500 real and 2500 fake).

The hyperparameters that have been selected to be optimized are: weight decay, learning rate and step size (only during the fine phase: because more epochs are needed). Also, attention was focused on using three different types of optimizers, to understand which of them was better in our case. The optimizers chosen are: Adam, RMSprop and SGD with momentum.

6.2. Coarse search results

In the coarse phase, each hyperparameter was sampled from these distributions:

Weight decay $\sim U(10^{-1}, 10^{-5})$;

Learning rate $\sim U(10^{-3}, 10^{-6})$.

The best results, obtained by each type of optimizer examined, during the approximate searching phase are shown in Table 1.

Given our evaluation method, taking the average between the accuracy and the maximum F1-score, this leads us in all three cases to always choose the model with the parameters that allow obtaining the highest accuracy and F1-score.

	Adam	RMSprop
Lr	0.0003	0.0004
WD	0.00003	0.00004
Step size	9	13
Accuracy	0.8834	0.8888
F1-score	0.6268	0.6155

Table 2: Best results of fine phase

	Accuracy	F1-score
Adam	0.6390 ± 0.0115	0.4855 ± 0.0052
RMSprop	0.6363 ± 0.0124	0.4848 ± 0.0069
SGD	0.5076 ± 0.0013	0.4023 ± 0.0011

This analysis leads us to say that RMSprop manages to achieve higher accuracy and F1-score values compared to the other two optimizers.

6.3. Fine search results

The results obtained from the previous phase can lead to choosing a subset of the range on which the sampling is to be carried out. From the results obtained, it can be seen that the SGD optimizer gets worse improvements than the other two. Therefore only Adam and RMSprop optimizers have been included for the fine phase.

So in this phase, we get this distribution:

Weight decay $\sim U(10^{-3}, 10^{-4})$;

Learning rate $\sim U(10^{-3}, 10^{-5})$;

Step size $\sim U(5, 15)$.

In the case of fine searching, the results are available in Table 2.

In the case of the Adam optimizer, the evaluation mode went to select the hyperparameter set with the highest value of the F1-score and not the maximum accuracy value. Instead in the case of RMSprop, it is different, the evaluation model favours accuracy and does not go to take the maximum F1-score.

	Accuracy	F1-score
Adam	0.8659 ± 0.0002	0.6145 ± 0.0001
RMSprop	0.8517 ± 0.0004	0.6028 ± 0.0001

In this the results are comparable, but it can be said that the Adam optimizer manages to get better results.

Furthermore, it should be noticed that in this case of search, no type of freezing of the network was made and we did not manipulate the data with transformations.

6.4. Training attempts

In order to improve our performance on classification, different training approaches were tried. At each run, over a certain number of epochs tried, only the best performing one was chosen.

Few attempts with 10000 videos (5000 fake and 5000 real) were tried, in order to catch the best training pipeline. The first one was a training with Adam optimizer and fine tuned parameters for 30 epochs. The 25th resulted to be the best one, with 88,64% of accuracy and 62,98% of F1-score on validation. However, 25 epochs seemed to be too much and the other initial attempts were tried with only 5 epochs, also to rapidly obtain comparable results.

Both Adam and RMSprop optimizers were experimented with their respective fine tuned parameters. These were their best results:

	Accuracy	F1-score	at epoch
Adam	0.7946	0.5277	3
RMSprop	0.7996	0.6195	4

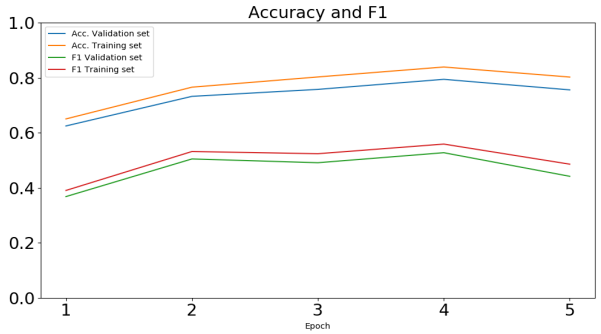

Figure 4: Training with 10k videos, Adam optimizer and fine tuned hyperparameters on FaceForensics++ weighs

Then, the field of image transformations was briefly explored. In particular, random white noise was tried, but results were slightly worse then previews ones, especially on accuracy (compare Figure 4 and 5).

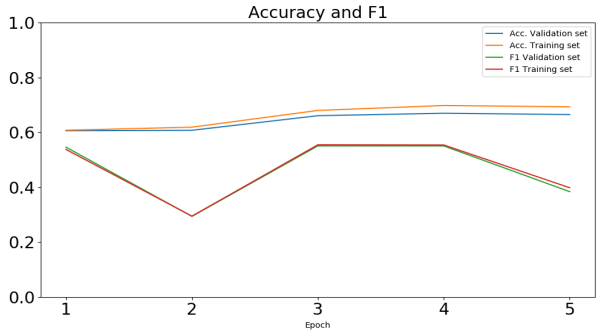

Figure 5: Training with 10k videos, Adam optimizer, fine tuned hyperparameters and random white noise on FaceForensics++ weights

After that, dataset manipulation methods were used to overcome the problem of highly imbalanced classes.

The first method tried was random under-sampling: a lot of fake videos were virtually randomly removed from the training dataset at each training, obtaining a number of fake videos equal to the amount of real videos. To be precise, 18663 samples for each class were kept. The best results were obtained training for 18 epochs, with very good resulting accuracies, around 94% on validation. Also F1-score seemed quite good, exceeding previous attempts, with a value around 65%. Then, transformation with random white noise was again tried, but results were again slightly worse, with an accuracy of around 87% and an F1-score of around 62% at epoch 19.

6.5. Kaggle's results

On Kaggle, Adam and RMSprop performed almost equally against Public Test Set. Despite our results on validation set, random noise transformation sometimes helped with predictions, and gave us one of our best results: 0.69609 as Kaggle computed score. Under-sampling did not perform well against the test dataset. More considerations on that are provided in Conclusions paragraph.

7. Transfer learning on ImagenNet weights

7.1. Coarse parameter search on ImageNet pre-trained model

We proceeded with the search of the hyperparameters with the pre-trained network on Imagenet, trying to emulate the methods carried out by the paper of FaceForensics++.

In this case, due to lack of time, we only provided coarse learning. Since wanting to try to decrease the competition score as much as possible, we decided to look for the best hyperparameters using the following division:

Training set: given by 134134 videos (67067 real and 67067 fake)

Validation set: given by 57488 videos (28744 real and 28744 fake)

To obtain these sets, we used a second dataset manipulation method: random over-sampling. The fact that at each step only a random frame of a video is chosen, made us confident to simply randomly duplicate real videos, to reach the amount of fake ones. However, with a total number of 191622 videos, we were forced to decrease the number of iterations to 10. Furthermore, network freezing was applied, in order to allow an higher batch size and reduce computing time.

In this phase, hyperparameters were sampled from these distributions:

Weight decay $\sim U(10^{-1}, 10^{-5})$;

Learning rate $\sim U(10^{-1}, 10^{-3})$;

The research was carried out only on one type of optimizer, in this case Adam.

	Adam
Lr	0.0060
WD	0.09915
Accuracy	0.5011
F1-score	0.6640

With only 5 epochs of coarse learning, accuracy never exceeded 53%, however an interesting F1-score of around 66% was obtained.

7.2. Training attempts

With ImageNet weights, under-sampling was directly selected as starting point. As for FaceForensics++ weights, F1-score was around 65%, but a slighty better accuracy was obtained: around 95% on validation at epoch 19. This was our best result against our validation set (Figure 6).

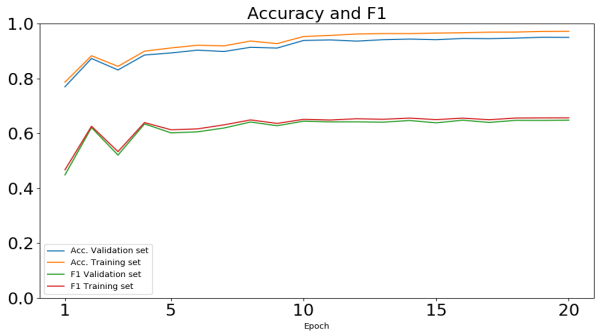

Figure 6: Training with under-sampling, Adam optimizer and fine tuned hyperparameters on ImageNet weights

Finally, we experimented over-sampling also on training. We had the opportunity to try it only with few hyperparameter sets and a couple of different numbers of epochs, but none of them succeded. Accuracy never exceeded 55% and F1-score had a fluctuating behaviour.

7.3. Kaggle's results

Even if our last over-sampling attempts did not give significant results, we obtained our best score on Kaggle (0.69484) with the hyperparameters found with the fine search on FaceForensics++ weights and freezing the whole network except for the fully connected layer while training from ImageNet weights.

8. Kaggle Notebook

On Kaggle we had a run-time limited to 9 hours on GPU for our submissions, so we used FaceNet, a face detector based on PyTorch, instead of dlib.

We trained our models offline, so we created a dataset in which our trained models were uploaded.

To make a prediction on a video, a certain number of equally distributed frames is extracted from it. Resulting frames are grouped in batches; for each batch, the face detector is applied and one of our trained models predicts how likely it is that detected faces are fake. Finally, for each video, the average on all predictions is assumed as the probability of the entire video to be fake. In case of errors like exceptions or no faces found, a probability of 0.5 is assigned to the video.

9. Conclusions

Based on the results obtained we can say that it is possible to improve our approach. One technique to do this, is to perform the manipulations made on the Public Test Set described in the Dataset paragraph, on the videos of the Training Set, before extracting frames. Furthermore, we can execute other types of transformations on frames to improve the training.

Another possible betterment is to use the FaceNet network also on the Training Set, to obtain more faces and in a shorter time. FaceNet is much faster than the dlib library, because it takes advantage of the parallelism of GPU, and performs well also on different poses (e.g. in profile).

Also, an hardware improvement could lead us to have more results in the same amount of time. We had a limited GPU-dedicated memory of 5GB and we had to work with very low batch sizes. But, fortunately, we gained some credits on AWS and, in future, we could have the possibility to speed up our research. It would be important also because we thought that a new validation method, more similar to how predictions are made on Kaggle Notebook, could lead us to more reliable evaluation metrics, but it would take much more computing effort.

Another key point of analysis may be to try different pretrained weights (e.g. raw, c40 from FaceForensics++) or a completely different network (e.g. ResNet) to see if with other levels of compression or a totally different structure it is possible to obtain more precise results against Kaggle Public Test Set.

References

[1] "You Won't Believe What Obama Says In This Video!". YouTube video, 1.12. Posted by "BuzzFeedVideo" April 17, 2018, https://www.youtube.com/watch?v=cQ54GDm1eL0.

[2] L. Verdoliva C. Riess J. Thies A. Rossler, D. Cozzolino and M. Nießner, 2019. FaceForensics++: Learning to Detect Manipulated Facial Images, Available: https://arxiv.org/pdf/1901.08971.pdf.

[3] B. Pflaum N. Baram C. C. Ferrer AI Red Team B. Dolhansky, R. Howes and Facebook AI, 2019. The Deepfake Detection Challenge (DFDC) Preview Dataset, Available: https://arxiv.org/pdf/1910.08854.pdf.

[4] Microsoft the Partnership on AI's Media Integrity Steering Committee AWS, Facebook. Data Description of Deepfake Detection Challenge, Available: https://www.kaggle.com/c/deepfake-detection-challenge/overview/getting-started.

[5] https://github.com/ondyari/FaceForensics.

[6] F. Chollet, 2017. Xception: Deep Learning with Depthwise Separable Convolutions, Available: https://arxiv.org/pdf/1610.02357.pdf.