Deep Learning Project - Deep Fake Detection

Nuno Correia, 58638
Faculdade de Ciências da Universidade de Lisboa
fc58638@alunos.fc.ul.pt

Daniel Neves, 64504
Faculdade de Ciências da Universidade de Lisboa
fc64504@alunos.fc.ul.pt

Abstract

This project investigates how design choices influence the classification of deepfaked videos. To evaluate this, we conducted multiple experiments with varying architectural and preprocessing strategies, repeating each configuration five times to ensure robust results. Our findings highlight the critical role of temporal cues in detecting deepfakes, with our best temporal model achieving an AUROC of 0.96 and an accuracy of 0.90. We also propose potential enhancements to improve performance further. Notably, our results suggest that increasing video frame quality may reduce the model's ability to generalize, underscoring a trade-off between image fidelity and classification robustness.

1. Introduction

Our project focuses on classifying videos as either real or deepfaked. Detecting deepfake videos is a particularly relevant challenge today, as the technology behind them continues to advance, making it increasingly difficult to distinguish fake content from genuine footage. Given the potential harm that deepfakes can cause ranging from misinformation to reputational damage, we were motivated to explore this topic for our project.

1.1. The dataset

The data used for this project was gathered from the Deep Fake Detection Challenge (DFDC) in Kaggle. It's a competition about building models to identify real videos from deep faked ones. The challenge can be found at https://www.kaggle.com/competitions/deepfake-detection-challenge/data.

1.2. Data pipeling and preprocessing

We began by creating a script called prepare_faces.py, which downloads videos from the Kaggle challenge page. Using OpenCV, the script extracts 10 evenly spaced frames from each video and attempts to detect and crop a face in each frame. If no face is found in a video, that video is discarded. For videos where a face is successfully detected, the 10 cropped frames are stored in a 4D tensor of shape (10, 3, 224, 224) and saved as a .npz file.

This process was repeated until we had collected 2,508 .npz files. These files represent 80% of the full dataset. To complete the dataset split, we added 314 samples for validation and 314 for testing, each representing 10% of the total dataset. This results in a final dataset split of 80/10/10 for training, validation, and testing, respectively.

Importantly, each subset of the dataset maintains a 50/50 class distribution, ensuring there is no class imbalance. In total, we extracted 3,136 videos containing at least one person. From each video, 10 evenly spaced frames were processed, and faces were cropped using OpenCV. While we initially considered using RetinaFace for face detection, it was ultimately excluded due to its significantly higher computational cost compared to OpenCV.

1.3. Setup and execution

The experiments for this project were executed on a personal computer and on Google Colab's T4 GPUs. This means that we did not consider run times for the different models because the disparity in time between the personal computer and the T4s was very big. For reference's sake the personal computer had the following specs:

- NVIDIA Geforce GTX 4080 Super
- Intel i9-13900KF
- 32 GB of RAM

This time disparity happened due to the virtual machines in Google Colab having way weaker specs when compared to the computer mentioned above.

2. Methodology and Experiments

We conducted 4 different experiments to better understand how design choices would affect our models when it came to their performance.

2.1. Methodology

2.1.1. Dataset augmentations

To ensure that the results and experiments were consistent and comparable, we kept the same augmentations across all of our 4 different experiments. The augmentations used were the following:

- Random Horizontal Flip(p=0.5)
- Gaussian Blur(kernel_size=3, sigma=(0.1, 1.5))
- Color Jitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.0)
- Random Resized Crop(224, scale=(0.8, 1.0), ratio=(0.9, 1.1))
- Random Erasing(p=0.7, scale=(0.02, 0.1), ratio=(0.3, 3.3))
- Normalization(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

Some reasons as to why we used these augmentations:

- Random Horizontal Flip, Gaussian Blur and Color Jitter were used to emulate social media re-encodings and camera noise.
- Random Resized Crop and Random Erasing were used to make the model less dependent on certain features, disincentivizing it from memorizing faces.
- Normalization was used so that the frames were in the format that the XceptionNet and the EfficientNet-B3 are expecting.

2.1.2. Hyperparameters

We kept some aspects the same across all experiments, such as the scheduler, the optimizer, the decision threshold, the batch size and the number of training epochs. The hyperparameters we used across the experiments were also kept the same. Such are the following aspects and parameters:

- The AdamW optimizer with a learning rate of $3E^{-4}$ and betas (0.9, 0.999).
- The CosineAnnealing scheduler.
- A batch size of 32.
- 10 training epochs.
- A decision threshold of 0.6.

The values for these parameters and the choice of scheduler and optimizer were found in the fine-tuning phase of the project and kept the same until the end.

2.2. Training and Testing Methods

2.2.1. Training set

Our Baseline model for the experiments is an XceptionNet with ImageNet weights, with 18 of the last unfrozen layers. We used the AdamW optimizer with the Coisine Annealing LR scheduler. We used the augmentations described in section 2.1.1 only in our training set across all experiments.

2.2.2. Testing and validation sets

For the testing and validating sets, we only normalized the images for the XceptionNet. After letting the model train for 10 epochs. At the end of every epoch we tested the model in the validation set and we registered the best performing model, in terms of AUROC. This way we ensured that for testing we used the best performing model and also as a way to avoid using a model that overfitted the training data. We saved the state of the best performing model's weights by creating a checkpoint.

After the 10 epochs we used the checkpointed model weights and we loaded them in our model and we tested the model on the training set.

2.3. Experiments

2.3.1. Experimental setup

To better compare the performance of the models with each other, we repeated each experiment 5 times and computed the mean value for all the metrics we considered across the 5 runs of each experiment. As well as registering the best run's values and bootstrap confidence intervals.

The metric we considered were the following:

- AUROC
- F1
- Accuracy
- EER

We used AUROC as the metric that defined our best model in validation. This is due to the fact that AUROC is the standard metric used across deep fake literature.

2.3.2. Experiments performed

The 5 experiments we performed were the following:

1. Using an XceptionNet as our baseline.
2. Using JPEG Quality Augmentation on our baseline XceptionNet.
3. Using Focal Loss on our baseline XceptionNet.
4. Using a stronger backbone, namely EfficientNet-B3.
5. Using an LSTM Head to our XceptionNet baseline model.

We'll now go more in depth as to why we decided to perform these experiments.

2.3.3. XceptionNet baseline

We chose an XceptionNet with ImageNet weights as our baseline model since it is state of the art in the Kaggle challenge we obtained our data from. Making it a good point of comparison with other models developed for this same challenge.

2.3.4. JPEG Quality Augmentation

With this experiment we wanted to check whether enhancing the videos' frames could also enhance artifacts that were produced by the insertion of deep fakes on the videos. Therefore, to do this we added to our augmentation pipeline in the training set a JPEG Quality Augmentation with a quality range of (10, 50).

2.3.5. Focal Loss

Since label smoothing, may hide hard positives, we used Focal Loss instead of BSE + 0.005. By using focal loss we wanted to see if we could make our model perform better with tricky deep fakes, even though we have balanced 50/50 sets we still thought that using focal loss could benefit our model. To test whether focal loss could help, we used focal loss with $\alpha = 0.25$ and $\gamma = 2.0$.

2.3.6. EfficeintNet-B3

To help us understand if a stronger backbone would help, we used an EfficientNet-B3 model. This model is more efficient and uses less parameters than our XceptionNet baseline. One important thing worth nothing on this experiment is that to unfreeze the 18 layers that we keep consistent throughout all experiments we had to perform gradual unfreezing. This is a limitation we encountered since if we immediately unfroze at the beginning it would take up all of the GPU's RAM, so the help alleviate this we gradually unfroze the backbone.

2.3.7. LSTM Head

Due to the fact that we are dealing with videos, we wanted to test whether temporal queues were important for the detection of deep fakes. Could there be artifacts between the frames' transitions that are important for classifying video correctly? To help us answer this question we passed each frame from a video through the baseline XceptionNet and then we fed the resulting vector to a Bi-LSTM. And we obtained a logit for a video from it.

2.4. Results

2.4.1. Result tables

The results we will describe in this section were obtained by running the experiments as described in 2.3.1. Below we provide four tables documenting the results we obtained, each table represents the values obtained in our four metrics that we discussed above.

Experiment	Mean	Best	CI (Best)
Baseline XceptionNet	0.86	0.93	(0.904, 0.959)
Baseline + JPEG Augmentation	0.81	0.85	(0.803, 0.888)
Baseline + Focal Loss	0.85	0.89	(0.855, 0.923)
EfficientNet B3	0.87	0.90	(0.857, 0.928)
Baseline + LSTM Head	0.92	0.96	(0.931, 0.975)

Table 1: AUROC results for different experiments.

Experiment	Mean	Best	CI (Best)
Baseline XceptionNet	0.76	0.88	(0.838, 0.911)
Baseline + JPEG Augmentation	0.71	0.76	(0.707, 0.803)
Baseline + Focal Loss	0.76	0.81	(0.764, 0.850)
EfficientNet B3	0.74	0.82	(0.780, 0.860)
Baseline + LSTM Head	0.83	0.90	(0.866, 0.933)

Table 2: Accuracy results for different experiments.

Experiment	Mean	Best	CI (Best)
Baseline XceptionNet	0.77	0.88	(0.844, 0.917)
Baseline + JPEG Augmentation	0.71	0.74	(0.685, 0.786)
Baseline + Focal Loss	0.71	0.80	(0.743, 0.850)
EfficientNet B3	0.69	0.82	(0.768, 0.857)
Baseline + LSTM Head	0.84	0.91	(0.870, 0.937)

Table 3: F1 results for different experiments.

Experiment	Mean EER	Best EER
Baseline XceptionNet	0.22	0.13
Baseline + JPEG Augmentation	0.25	0.25
Baseline + Focal Loss	0.25	0.20
EfficientNet B3	0.21	0.17
Baseline + LSTM Head	0.15	0.09

Table 4: EER results for different experiments.

From table 1, we can see that the model which obtained the best performance in terms of AUROC was our baseline XceptionNet with an LSTM Head. The EfficientNet-B3 also performed well overall and so did our baseline XceptionNet. The introduction of JPEG Quality Augmentation actually made the baseline's AUROC go down.

This same pattern of our baseline XceptionNet with an LSTM head being the best model repeats across all of our experiments in all other metrics. The only exception to this is that for the F1 metric the EfficientNet-B3 was actually the worst one. This is due to the fact that it had a particularly bad run in one of the experiments. We believe that if we repeated the experiment 5 more times its F1 would also follow the same trend as the other metrics.

2.4.2. Learning Curve of the Best Model

Our best model, in terms of AUROC, was the LSTM Head. The graph below shows the evolution of the AUROC curve in the validation set. The validation AUROC was still increasing when we stopped at the 10th epoch. This means that had we trained this model for 5 more epochs we can maybe squeeze some more AUROC from it.

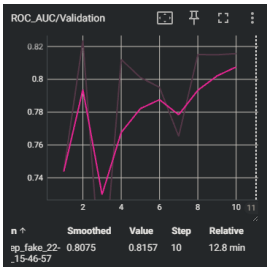

Figure 1. Learning Curve of the best model on validation

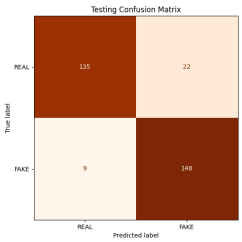

Figure 2. Confusion matrix of the best model

From the confusion matrix we can see that our best model performed very well in classifying FAKE videos. As it barely got any FAKE videos wrong.

2.4.3. Best Model

The LSTM Head experiment produced the highest AUROC due to its temporal aspect; The EfficientNet B3 experiment produced less variation in the AUROC results due to its stronger backbone, the JPEG Quality Augmentation experiment reduced the AUROC value, possibly due to much augmentation. The Focal Loss experiment produced a lower EER compared to the JPEG Quality Augmentation and the EfficientNet B3 experiments.

2.4.4. Focal Loss Results

One very interesting result we got from the focal loss experiment runs is that on average the focal loss runs outperformed our best model on classifying REAL videos. On average the focal loss experiments only got 14.4 REAL videos wrong and got 142.6 REAL videos right. However, on FAKE videos it performed way worse than the LSTM head. This means that maybe using focal loss with the baseline with the LSTM head might yield better results in the classification of REAL videos. This can be seen on the confusion matrices from the 5 runs of the focal loss experiment.

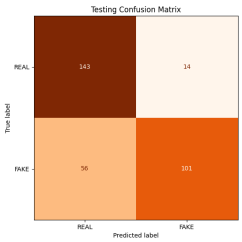

Figure 3. 1st run focal loss results on testing

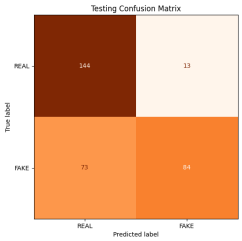

Figure 4. 2nd run focal loss results on testing

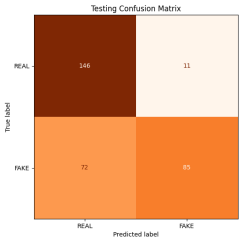

Figure 5. 3rd run focal loss results on testing

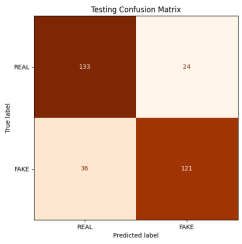

Figure 6. 4th run focal loss results on testing

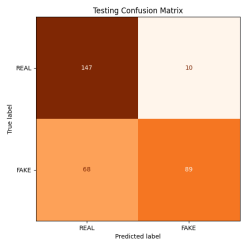

Figure 7. 5th run focal loss results on testing

As we can see on the confusion matrices, the baseline Xception model performed quite well when classifying REAL videos in particular.

2.4.5. Conclusions from the results

We can conclude that temporal cues matter a lot in the classification of deep fake images. This can be seen in our results as the LSTM Head experiment was by far the best and the one that showed the least variance in the AUROC CI. And also as seen on the graph in image 1, the validation kept increasing up to the 10th epoch. We could probably squeeze a better AUROC value if we continue for 5 more epochs. Using focal loss with the baseline XceptionNet and with an LSTM head might also give us an increase in AUROC.

2.4.6. GradCam

We produced GradCam activation heatmaps for the last convolution layer on our best performing model (XceptionNet + LSTM Head). The images below (8 and 9) are both correctly classified examples from the test set. For frames predicted as REAL there are usually no activation on the face. As for frames predicted as fake, there are activations typically around the eyes, nose and mouth. These patterns were verified a couple of times however, there are some cases where an image is classified as REAL and it has activations on the face.

Figure 8. Correctly classified FAKE

Figure 9. Correctly classified REAL

3. Conclusions

To conclude, our main takeaway from this project is: Temporal features are more important than image-level detail for this task. Future improvements to this model could be training for 5 more epochs and trying to use focal loss with our best model due to its good performance on classifying REAL videos. We could also attempt to test our best model with video generated by AI or with more modern videos to see if it can detect more modern deep fake videos or not. That is, do deep fake videos still possibly generate the same artifacts as they did before but we just can't see them with our naked eyes? Overall, this is a topic with a lot of possibilities when it comes to exploring it further.