Identifying Deepfake Videos with Convolutional Neural Networks

Zhixing Che and Julian Ingham
Boston University
Commonwealth Avenue, Boston, MA 02215, USA

Abstract

In this project we explored the task of discriminating between real video data and Deepfakes. We processed a large volume of video data into a set of still color images. We applied a convolutional neural network to detect artifacts of the forgery in the facial region, such as distortions and color blotches. We designed our own network architecture, experimenting with various parameters including network depth, dropout rate, and batchsize, among others. Our best results on our test set were highly accurate, among the top 100 in the Kaggle Deepfake Detection Challenge (out of 2,265 entrants). Further, we found that certain kinds of data augmentation decrease predictive power, suggesting the presence of asymmetric correlations in Deepfake artifacts. We summarize preliminary additional experimentation with transfer learning.

1 Introduction

Deepfakes are synthetic media, in which the source images or videos have been manipulated to emulate likeliness to another person, or to create factitious characters. The name "Deepfake" comes from "Deep learning" and "fake." Recent advances in Generative Adversarial Networks (GANs) have made Deepfakes as realistic and deceptive as ever. GANs synthesize images through pitting a generative network and an discriminative network against each other, competing to produce an iteratively more compelling fake, and their output is increasingly hard to distinguish from real videos.

Deepfakes clearly threaten public discourse, to the extent they can produce compelling fake statements by public figures. In general, they also allow the identity of public figures to be co-opted in insidious ways. There is therefore a strong motivation to create precise and automated detectors of Deepfake videos. Some highly publicized examples include a Buzzfeed production of a Deepfake of President Obama, depicting him insulting President Trump (1), as well as a video of Mark Zuckerberg saying that his true mission is to manipulate Facebook users (2).

In light of the timeliness of this topic, Kaggle held a Deepfake detection competition earlier this year, in collaboration with Facebook (3). A dataset of approximately 470 Gb of video data was made publicly available, consisting of real and Deepfake videos produced specifically for the competition. 2265 teams competed, and were ranked based on loss performance both on the public dataset as well as a private, unreleased dataset. The lowest loss achieved on the public dataset was 0.19207; in spite of the encouraging results obtained on the publicly available dataset, the results for the private dataset were considerably worse, illustrating that current Deepfake models struggle to generalize well.

Sophisticated Deepfakes are a relatively recent phenomenon, but there are already several serious academic studies of the topic. State of the art techniques can be roughly broken down into two directions: emphasis on classifying facial expression signatures of notable individuals, and emphasis on finding inconsistencies exhibited in the physical/physiological aspects of the Deepfake. The former typically make use of Support Vector Machines, with facial analysis data. S. Agarwal et al. focused on specific politicians – who are more likely to be the subject of Deepfakes – training on

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

real examples of those individuals' speech and gestures (5). The shortcomings of this approach is that it is highly specialized to the limited number individuals trained on, and therefore limits possible applications. It also sheds little insight into the structural characteristics of Deepfakes more generally. The second main research direction focuses on finding artifacts of the synthesis introduced by the GAN – such as over-blurred features, discoloration, etc. Y. Li, et al. combined a convolutional neural network (CNN) with a recurrent neural network, to analyze sequential frames and detect whether the video subject was blinking fewer times than is normal for an actual human (4). However, recent advances in Deepfakes have incorporated blinking into their systems so that this technique is now less effective.

1.1 Motivation

Deepfakes impose a face onto a head in a video. Since this process is not perfect, it is natural to expect that Deepfakes possess visual defects in the facial region. We therefore imagine that while most of the video is real video, focusing on the facial region in a Deepfake gives us a good signal of real or fake video.

Some exploration of this idea is present in the existing literature on this topic. For instance, U. A. Ciftci et al. investigated the classification of Deepfakes through the observation of biological signals in the facial region (6). First learning to extract specific biological signals, a representation of those signals is fed into a CNN, which then classifies a video as fake or real. This method appears to be the current state of the art as far as our literature review could find, however, as the Kaggle challenge illustrates, it is not necessarily clear how well high performing models generalize across different datasets.

1.2 Background: Convolutional Neural Nets

A convolutional neural network (CNN) is a way to exploit symmetries and invariances in a dataset, by applying filters to a dataset which naturally detect local features. A convolutional layer detects variations in a local region of the input, and scans across input to identify the locations and structure of those variations. The technique draws inspiration from biology – neurons of the visual cortex are organized such that some neurons respond only to stimuli in a restricted region of the visual field.

Since the defects introduced by the superposition of two faces in a Deepfake are local, and probably exhibit spatial patterns in particular regions of the face, a CNN is a natural classifier for exploiting the two dimensional structure of our data.

2 Dataset

The dataset we relied on was drawn from the Kaggle Deepfake Detection Challenge, run by Facebook and Google. The organizers paid a group of actors to record thousands of videos, and a large number of Deepfake videos were produced using this material. The dataset is available on Kaggle in 10 Gb chunks. We downloaded a total of 5 chunks, amounting to 13122 videos.

The obvious challenge we immediately met with was the sheer size of the data we were working with. Not only are large files difficult to save to memory and manipulate, but they also contain a great deal of irrelevant information (the background of the video is unlikely to contain signatures of a forgery). In order to load the videos to manipulate, we wrote code which split the total set of videos up into manageable "chunks", which we then loaded individually. To isolate signals from noise, and to format our data into a manageable size, we wrote code which took each video and extracted 5 random frames, identified and extracted the face in each frame, resized the resulting image into a $252 \times 252 \times 3$ image, and saved the faces and associated labels (0 for fake and 1 for real) as .npy files. In Fig. 1, we show one example of a frame taken from a Deepfake video, alongside the resized face which our code produced from it.

Early on, we also noticed that the datasets provided by Kaggle were strongly imbalanced towards fakes. Around 20 % of each data folder are real videos, and the rest are fake videos created from the real ones. Rather than deal with the subtleties of training on imbalanced data, we downloaded a large volume of the videos, and saved a balanced subset of the total data points.

Figure 1: An example of a still frame extracted from one of the videos, alongside the facial region we extracted from it. Remarkably, this is in fact an example of a Deepfake.

Finally, we produced a total collection of 17646 images. We initially took a set of 10278 pictures, which were split into a training set of size 7364, validation set of 1426 and test set of 1488. We were so surprised with the high accuracy of the network we designed, that we processed an additional 7368 images from 20 Gb of totally different Deepfake videos – as a test of how well our model could generalize to totally new data.

As we will discuss later, remarkably, we found our model performed only negligibly worse on the entire second dataset, proving our model is exceptionally robust in addition to highly accurate. While the accuracies we obtained are still worse than state of the art results presented in the literature, such robustness appears unlike any model we could find. Comprehensively testing this discovery is an important future research direction.

In order to easily load the large number of images we saved, we saved our images to SCC and constructed our train, validation, and test sets using the keras preprocessing iterator `ImageDataGen()`. This command also allows us to naturally implement data augmentation by inputting arguments such as horizontal flip, or rotation angle range.

3 Results

We began by designing our own network from scratch. Through trial and error, we made the following observations:

- Working on small datasets (on the order of the size of only one folder) generally failed to yield good results. Furthermore, the way data was segmented in each folder (only a few actors in each one) is likely to attribute to the model overfitting to irrelevant details, thus being unable to generalize for other data points it has not seen before.
- Beginning with a network with a large number of nodes and large dense layers (with $200 - 500$ nodes each layer), we initially found a significant (as large as 0.15) difference between the training and testing error. This is intuitively attributed to overfitting.
- We saw an improvement about adjusting the depth-wise structure of the network. By making the initial convolutional layers larger, with larger filters and stride, and narrowing the network in subsequent layers, we also found an increase in training and testing accuracy. This is also intuitively explained as facilitating the network identifying coarse-grained features before fine-scale details.

We were initially pessimistic about the possibility of decent accuracy, since our early models generalized poorly to new data. Our main breakthroughs came from massively increasing the size of our dataset, shuffling training data to avoid training for irrelevant features, and significantly pruning the size of our initial network, reducing the number of nodes substantially. After these changes, the typical reported accuracies immediately jumped by $\sim 10\%$.

3.1 Network architecture

In the subsequent discussion, the results we present are based off the architecture in Fig. 2, which seemed to overall do the best out of our experiments.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 86, 86, 75)        3675
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 43, 43, 50)        33800
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 21, 21, 50)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 21, 21, 50)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 25)        5025
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 25)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 10, 10, 25)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2500)              0
_________________________________________________________________
dense_1 (Dense)              (None, 200)               500200
_________________________________________________________________
dropout_3 (Dropout)          (None, 200)               0
_________________________________________________________________
dense_2 (Dense)              (None, 100)               20100
_________________________________________________________________
dropout_4 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 101
=================================================================
Total params: 562,901
Trainable params: 562,901
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Summary of the architecture of our CNN.

We trained the model several times, and found there to be some variance in the rate at which the network converged to optimal performance, the results below are the most recent training run we performed.

3.2 Benchmarking performance with human accuracy

Human performance (typically that of an expert) is typically used to estimate the Bayes rate, or intrinsic difficulty of the problem [1], though clearly this particular classification problem is one which is very difficult for humans in general, and we did not have access to an expert. We gave a classmate of ours a balanced subset of the data to predict on; the results were an accuracy of 0.59, recall of 0.66, and precision of 0.53. Deepfake detection therefore appears to be an intrinsically hard problem.

3.3 Inference

Training on 7364 images, with a validation set of 1426 and test set of 1488, we were able to achieve a high score of 0.8412 test accuracy (all other predictive metrics were similar, and performance on the training set was only slightly better). Moreover, the best trained version of our model achieved a loss of 0.51, placing us narrowly around the 100th place mark in the Deepfake Detection Challenge on Kaggle. However, this statement should be qualified with the observation that we only measured the performance of our model on subset of the data the competitors were judged on (the data we extracted amounted to about 50 Gb out of the total 470 Gb). The results can be viewed in our submitted code, and are also summarized below.

At the end of this period, we achieved a training accuracy of 0.813, as shown in Fig. 3. In Fig. 4, we report the AUC, precision, and recall for the first 100 epochs. The resulting test accuracy was 0.814. The results on the test set are summarized by the confusion matrix and ROC curve in Fig. 5.

After training for a further 30 epochs, we achieved the highest scores we can confidently report (Test set: accuracy $= 0.8412$, precision $= 0.8650$, recall $= 0.8096$; Training set: accuracy $= 0.8416$, precision $= 0.8652$, recall $= 0.8098$).

To demonstrate that this is not simply a result of overfitting the data, we tested the model on the extra 7368 data points we uploaded later in the project – which were obtained from qualitatively different Deepfake videos, from completely separate folders on the Kaggle website, and featuring different

[1] c.f. the discussion in Andrew Ng's tutorial at the Deep Learning School, available at https://www.youtube.com/watch?v=F1ka6a13S9I.

Figure 3: The training and validation accuracy during the first 100 epochs of training.

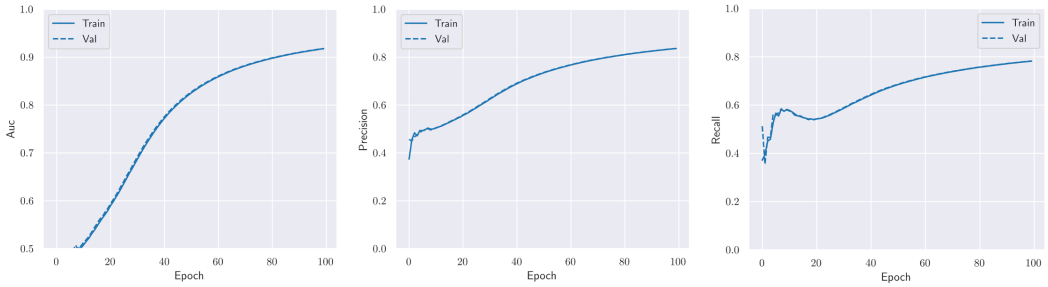

Figure 4: The AUC, precision and recall as a function of epoch for the first 100 epochs.

actors. We tested our network (with no additional training) on this extra data, found very similar performance (accuracy $= 0.8412$, precision $= 0.8646$, recall $= 0.8097$), providing a compelling demonstration of the remarkable robustness of our CNN [2].

3.4 Effects of data augmentation

Data augmentation refers to randomly transforming portions of the training data (through e.g. rotations, reflections, dilations), and is typically used to assist a CNN in identifying invariances in the dataset, thus increasing predictive power. The motivation is to help network to learn, for instance, that cats can sometimes be upside down or rotated even if your data are mostly pictures of cats standing upright.

However, we found that certain augmentations made prediction markedly worse. In particular, rotating the data seemed to lower the accuracy by as much as about 10 %. We conjecture the possible existence of systematic spatial patterns in the facial defects, which the network is taking advantage of to help learn, but are not rotationally invariant, such that they vanish upon averaging over rotated samples. For example, maybe the GANs which make Deepfakes find it harder to adhere the face onto the right and left side compared to the forehead and chin.

It is also reasonable to imagine that since the test set only contains upright faces, augmentation introduces invariances which are not present in the test set, reducing performance. To distinguish these two possibilities, we propose to create more data by selectively deleting portions of the face such as the jaw and forehead, resizing the image and testing performance of the network on this data without rotations. This would allow us to distinguish between the effects of having to learn the features of a sideways face, and learning from asymmetric defects. We did not have time to perform this experiment, but we think it would be a natural test to potentially shed valuable light onto the structure of Deepfake artifacts, and the inherent weaknesses of GANs.

[2] For completeness, and comparison with Figs 3 and 4, we also mention that the performance on the extra data after the first 100 epochs was accuracy $= 0.8135$, precision $= 0.8351$, recall $= 0.78167$).

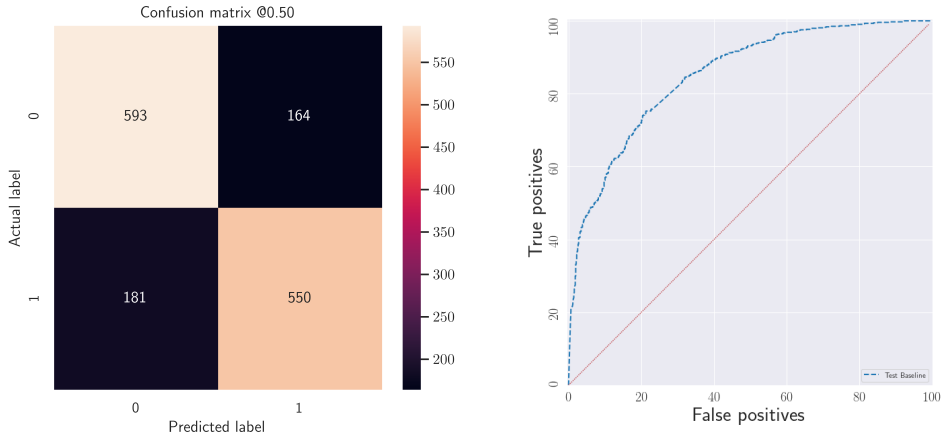

Figure 5: The confusion matrix and ROC curve after the first 100 epochs of training. The notation used for the confusion matrix is 0 for fake and 1 for real.

Other data augmentation methods that we experimented with includes zoom and horizontal flip. We avoided trying augmentations that could distort the facial features such as shears. Zoom with a range of 0.1 lowered training and testing accuracies by around 5%. We decided to omit it from our final model as the lower metrics suggest that crucial information on the edges of the face could be lost when zoomed. Including horizontal flip did not produce a significant effect on the model.

3.5 Transfer learning

A powerful tool in the use of neural networks is transfer learning – the method of downloading a previously trained network, and then adding additional layers. The idea is that the sophisticated pretrained neural network does a great job identifying broad trends, then the additional layers fine tune the output to identify problem-specific details.

Pre-trained networks such as Resnet can add to the sophistication of the image recognition, and networks such as Efficientnet can help achieve optimal performance through adjusting the depth and connectedness of the CNN.

Efficientnet builds upon the compound scaling method theory that adjusts depth, width and resolution based on a ratio (7). The theory lies behind the empirical observation that balancing the three parameters is critical for accuracy and efficiency when training a CNN.

We originally tried downloading `EfficientnetB0` and applied it to our training data. It started with strong performance, with accuracy in the low 0.8s, but the accuracy steadily decreased with epoch – losing 3% accuracy after 15 epochs. It is possible that the data we have – small and processed images of faces – are already too small and do not have that many features, so that being passed through a large pre-trained network denies the CNN a chance to identify the problem-specific features.

In future work, we would like to spend more time adjusting a pre-trained network to see if we can obtain superior accuracy.

4 Conclusion and Outlook

In this project we designed and implemented a CNN as a classfier to discriminate between a dataset we processed, corresponding to real and Deepfake videos. The main issues we encountered were creating a suitable dataset from high resolution videos, and designing our CNN to take advantage of a larger dataset. We conducted several experiments to identify the ideal architecture, and achieved relatively high accuracy, precision, and recall – ranking competitively alongside the public results of the Kaggle Deepfake Detection Challenge. We made some interesting observations about the effects

of data augmentation on the learning process, and some preliminary tests of the efficacy of some pretrained networks.

Additionally, as noted earlier, our model performed just as well on a new test set we created – of totally different Deepfake videos, and of magnitude greater even than the training set. The top result in the Kaggle competition scored approximately 15% worse on the unreleased dataset as on the public dataset. While we do not have access to the private dataset, we would like to create our own data which differs even more starkly from the Kaggle public data, and explore if the impressive robustness of our model holds in general.

In future work, we would like to further experiment with pre-trained networks like Densenet, Resnet, or other versions of Efficientnet, to see how this improves our performance. It is not necessarily obvious that these pre-trained networks would increase our performance, since presumably Deepfake detection requires the identification of very different features to those ordinarily targeted in image recognition. Additionally, we would like to conduct much more rigorous experimentation on the precise effects of data augmentation, and what they can tell us about the underlying correlations in Deepfakes. Finally, it remains to be seen whether the architecture we converged on is optimal, or whether further experimentation could further improve our results.

5 Acknowledgements

We would like to thank Professor Chin, Peilun and Hieu for an enjoyable semester

References

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

[2] Posters, Bill. "I Wish I Could..." Instagram, 13 June 2019, https://www.instagram.com/p/BypkGIvFfGZ/, Last Accessed 2 July 2020.

[3] Deepfake Detection Challenge, https://www.kaggle.com/c/deepfake-detection-challenge/overview, Last Accessed 2 July 2020.

[4] Yuezun Li, Ming-Ching Chang, Siwei Lyu "In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking", [arXiv:1806.02877]

[5] S. Agarwal et al.,"Protecting World Leaders Against Deep Fakes", The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 38-45 (2019).

[6] U. A. Ciftci et al., "FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals", [arXiv:1901.02212]

[7] Mingxing Tan, Quoc V. Le,"EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", [arXiv:1905.11946]