Video deepfakes detection using Deep Learning

Dong Bing Luke Kim Sergei Petrov
{bingdong, mkim14, spetrov}@stanford.edu

Stanford

Motivation

Deepfake techniques, which present realistic AI-generated videos of people doing and saying fictional things, have the potential to have a significant impact on how people determine the legitimacy of information presented online. In this project we are focusing on using machine learning techniques to detect deepfakes using data provided by Kaggle as a part of a challenge [1]

Datasets

There are 4 groups of datasets available:

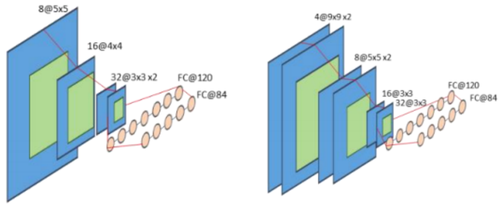
- Training set: The complete dataset, containing labels for the target. There are 470 Gb of archived videos in it.
- Sample sets: There is a small dataset of 400 labeled videos directly available from any notebook within the challenge, we used it for testing purposes. Also there is a set of 400 unlabeled videos that are used to create an output submission table.
- Public test set: This dataset is completely withheld and is what Kaggle's platform computes the public leaderboard against. When notebook is committed, the code is re-run in the background against this dataset.

Methods

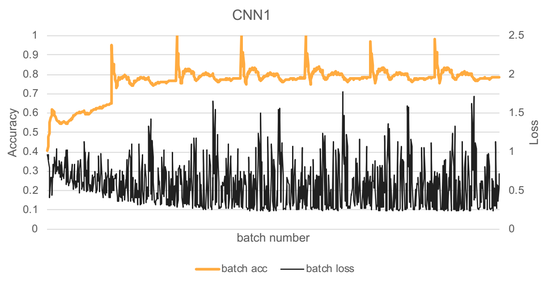
- **ResNet-50** and **VGG16** are deep convolutional neural network pretrained on more than a million images from the ImageNet repository.
- **CNN+LSTM** consists of several blocks of conv layers with an LSTM head. Each layer preceding an LSTM is time distributed, meaning that it was applied to every time step in an input example. Input to the network was 4-dimensional, with the 4th dimension corresponding to time [3].
- **Xception** architecture includes 36 convolutional layers that are structured into 14 modules all of which have linear residual connections around them, except for the first and last modules. The idea is based on depthwise separable convolutions usage. First, to each of the input channels 3x3 convolution is applied, extracting spatial information from each channel independently. Then, 1x1 convolution is applied to gain information from a cross-channel dimension [4].

Experiments and Implementation

- **Logistic Regression** and **SVM** (with linear kernel) were implemented. Regression yielded a train accuracy of **83.25%** and a validation accuracy of **83.33%**. SVM did worse than Logistic Regression, ending up with a **67%** validation accuracy.
- **ResNet50** and **VGG16** were finetuned on our dataset. Training accuracy reached **0.8**, validation accuracy was lower, around **0.7**.
- We trained simple **CNNs** from scratch. We tried two architectures shown in Figure 1. At the end of both networks there are two fully connected layers with 120 and 84 hidden units. The first architecture performed better. After the 1st epoch the network seems to make no progress, and its training accuracy reaches a plateau at around **0.8**. Validation accuracy reached **0.75**.

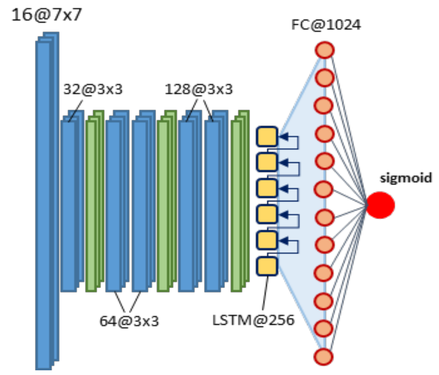


CNNs architectures

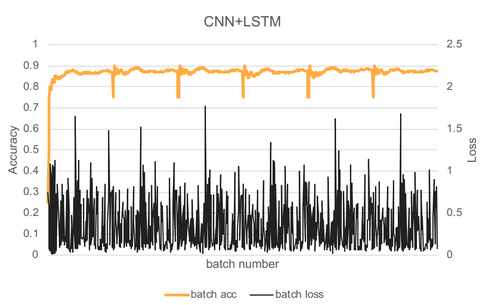


CNN training process

- We trained **CNN+LSTM** architecture. Again, training accuracy reaches plateau at around **0.87**, what is better, than for CNN architectures described above, but the algorithm still cannot reach a satisfactory accuracy value. Validation accuracy reached **0.85** at the end of training.

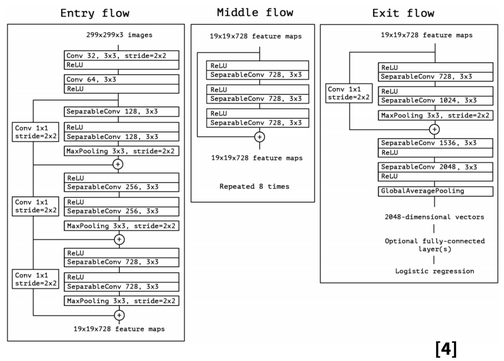


CNN+LSTM architecture



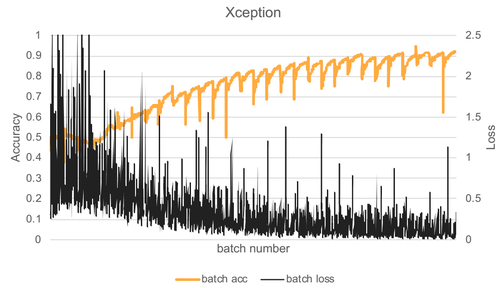
CNN+LSTM training process

- Our final experiment involved an **Xception** architecture. We reproduced the architecture from the original paper [4] and then tuned hyperparamets and played a bit with the architecture itself.



[4]

Xception architecture



Xception training process

With Xception we managed to fit the training set perfectly, the maximum validation accuracy achieved was around **0.84**. Also the validation loss is the lowest, it went down to **0.05** at the end of the training.

Discussion

- It seems that even deep conventional purely convolutional architectures are unable to capture the patterns within data to tell the difference between real and fakes with enough level of confidence
- Unsatisfactory accuracy achieved with LSTM It may be related to the sequential nature of LSTM – it is good at recognizing sequences (i.e. classifying different activities in videos), but any actions in our training examples are completely unrelated to their labels. Videos can be real or fakes regardless of whether actors move their heads, turn, walk or remain motionless.
- Xception showed the best results, but was still unable to reach human level accuracy. Probably, the source of the problem is challenging nature of task and complexity of the data

References

[1] Kaggle deepfake detection challenge https://www.kaggle.com/c/deepfake-detection-challenge
[2] Thanh Thi Nguyen, Cuong M. Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen and Saeid Nahavandi Deep Learning for Deepfakes Creation and Detection. https://arxiv.org/pdf/1909.11573.pdf
[3] https://github.com/harvitronix/five-video-classification-methods
[4] Francois Chollet. Xception: Deep Learning with Depthwise Separable Convolutions https://arxiv.org/pdf/1610.02357.pdf

.