DFDC Model Summary

A1. Background on you/your team

- Competition Name: Deepfake Detection Challenge
- Team Name: The Medics
- Private Leaderboard Score: 0.43711
- Private Leaderboard Place: ?

- Name: James Howard
- Location: London, United Kingdom
- Email: james@jph.am

- Name: Ian Pan
- Location: Providence, Rhode Island
- Email: ianpan358@gmail.com

A2. Background on you/your team

If part of a team, please answer these questions for each team member. For larger teams (3+), please give shorter responses.

James

- What is your academic/professional background?
    - I am a trainee cardiologist in London. I am currently taking three years out to undertake a PhD investigating the use of deep learning in cardiac imaging. I don't have formal programming or AI training, but I have published extensively in the field of cardiology and I have a strong understanding of statistics.

https://scholar.google.com/citations?user=CyuQ1RUAAAAJ&hl=en & https://james.dev/CV.pdf

- Did you have any prior experience that helped you succeed in this competition?
    - I have some experience of medical video classification using different convolutional neural network architectures: http://jmai.amegroups.com/article/view/5205/html
- What made you decide to enter this competition?
    - I had been aware of Kaggle competitions for some time, but had never got around to entering one. When I saw there was a competition involving video classification, I thought this would be a reasonable one to embark on as my first competition.
- How much time did you spend on the competition?
    - On average maybe 4 hours a day for the last 2 months of the competition. Towards the end of the competition we were running out of new ideas to try and so I was probably just spending an hour a day over the last two weeks, but earlier on it had been much more, especially when it came to pre-processing the videos.
- If part of a team, how did you decide to team up?
    - Ian contacted me around 6 weeks before the competition ended. Until that point I'd been reluctant to team up, despite many friendly Kagglers offering, as I felt my performance was strong and that people would just assume my partner had done most of the work as I was 'merely' a medic with no competition experience. However, the idea of pairing up with a fellow medic was really attractive, and Ian had lots of previous competition experience.
- If you competed as part of a team, who did what?
    - Before Ian joined I had developed the pre-processing pipeline which turned the original MP4s into smaller videos cropped down to distinct faces within videos, which we used to train 2D and 3D CNNs. I used this to train the 7 3D CNNs we used in our final solution. I had also tried several other networks including a few unsuccessful 3D CNNs, time-distributed CNNs, and some 2D CNNs, but these were inferior to Ian's models.

Ian

- What is your academic/professional background?

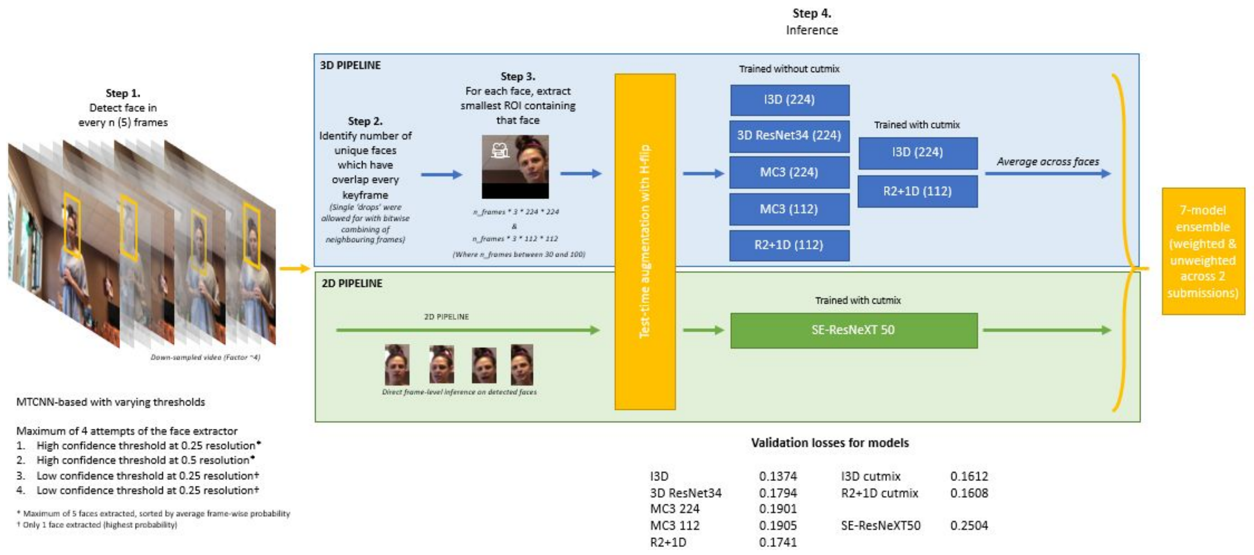
- I am a graduating US medical student entering radiology residency. I studied applied mathematics and statistics in college before medical school.

- Did you have any prior experience that helped you succeed in this competition?
    - My research over the past several years has been focused on applications of deep learning to medical imaging. I have participated in several other Kaggle competitions and earned gold medals in 2 competitions focused on chest X-rays.
- What made you decide to enter this competition?
    - This competition provided an opportunity to work with a large 3D dataset as opposed to other competitions that primarily focus on 2D images. Given that radiology has its fair share of 3D images, I felt that this would be a good opportunity to experiment with various approaches to 3D data. The problem of identifying deepfakes is also an important one in this new technological age.
- How much time did you spend on the competition?
    - I entered the competition about 6 weeks prior to its end. I spent about 20 hours/week on average reading relevant literature, writing code for experiments, and debugging. Most of the time was spent on debugging.
- If part of a team, how did you decide to team up?
    - James and I both came from a medical background, so we thought it would be cool to create a unique duo. At that time, I had just started experimenting with my 2D models and reached out to him. He was kind enough to accept my request.
- If you competed as part of a team, who did what?
    - I focused on creating a 2D model that would complement James' 3D ensemble. I also experimented with alternative face detection approaches and segmentation models, which ultimately were not part of our final solution. I experimented with many data augmentation approaches, primarily CutMix/Mixup.

A3. Summary

4-6 sentences summarizing the most important aspects of your model and analysis, such as:

Of our 8 models used in our final solution, 7 of them were 3-dimensional convolutional neural networks (3D CNNs) which were trained on videos cropped down to individual faces.


Step 1.
Detect face in every n (5) frames
Down-sampled video (Factor ~4)
3D PIPELINE
Step 2.
Identify number of unique faces which have overlap every keyframe
(Single 'drops' were allowed for with bitwise combining of neighbouring frames)
Step 3.
For each face, extract smallest ROI containing that face
n_frames * 3 * 224 * 224
&
n_frames * 3 * 112 * 112
(Where n_frames between 30 and 100)
Test-time augmentation with H-flip
Step 4.
Inference
Trained without cutmix
I3D (224)
3D ResNet34 (224)
MC3 (224)
MC3 (112)
R2+1D (112)
Trained with cutmix
I3D (224)
R2+1D (112)
Average across faces
2D PIPELINE
2D PIPELINE
Direct frame-level inference on detected faces
Trained with cutmix
SE-ResNeXT 50
7-model ensemble (weighted & unweighted across 2 submissions)

MTCNN-based with varying thresholds

Maximum of 4 attempts of the face extractor
1. High confidence threshold at 0.25 resolution*
2. High confidence threshold at 0.5 resolution*
3. Low confidence threshold at 0.25 resolution†
4. Low confidence threshold at 0.25 resolution†

* Maximum of 5 faces extracted, sorted by average frame-wise probability
† Only 1 face extracted (highest probability)

Validation losses for models

I3D	0.1374	I3D cutmix	0.1612
3D ResNet34	0.1794	R2+1D cutmix	0.1608
MC3 224	0.1901		
MC3 112	0.1905	SE-ResNeXT50	0.2504
R2+1D	0.1741		

These videos were generated using a face extractor pipeline which was written within the first week or so of starting the competition, with only fairly minor tweaks following. In summary, every nth frame (we settled on 10) is passed through a face detector (MTCNN). The bounding box coordinates are then used to set a mask in a 3D array. Faces which are contiguous (overlapping) in 3D are assumed to be a single person's face moving through face and time. We then extract a bounding box which includes this entire face over time and create a video from this region of interest, including every frame (not just every 10th frame). One nice thing about this method, even ignoring the video aspect, is it greatly reduced false positives, because the 'face' had to be present for a long period of time in the video to count as a face.

We both used the Pytorch framework to create and train our models. James and Ian worked separately on the 3D (top blue pathway of diagram) and 2D (bottom green pathway of diagram) pipelines, respectively, and so the code attached is organised in two folders accordingly.

A4. Features Selection / Engineering

We felt 3D CNNs would generalise well to 'different' methods of deepfakes. Instead of focusing on specific pixel patterns in frames, we hope they might identify temporal problems like the often tried-and-failed 2D CNN-RNN models, but with a much lower tendency to overfit. The 3D models have similar numbers of parameters to 2D models, and yet also the kernels have a 'depth', so they are relatively modest in their ability to fit in the 2D plane.

The 3D CNNs were trained on ~250,000 MP4 videos of >= 64 frames duration and 256 * 256 pixels, generated using the pipeline mentioned in the previous section.

We used no external data, other than pretrained (Kinetics for 3D, ImageNet for 2D) model weights which we have declared separately.

A5. Training Method(s)

Despite the task being a binary classification problem, the 3D CNNs were actually trained using a categorical (rather than binary) cross-entropy loss function. This is because one of the models (I3D) relies on having a convolutional kernel for each output class, meaning 2 outputs are required for a binary classification problem. The 2D CNN was trained using the binary cross-entropy loss function.

The input data (videos for the 3D CNNs, images for the 2D CNNs) were augmented during training in different ways depending on the model, but typically included gaussian blurring and noise, random brightness and contrast, random resized cropping and horizontal flipping. Two 3D models and the 2D model were also trained using CutMix. Test time augmentation with horizontal flipping was also used.

The networks were trained with AdamW (or sometimes Adam or Ranger) with a 1-cycle learning rate scheduler.

The final ensemble consisted of 8 models. We submitted 2 solutions based on the same ensemble: one was equally weighted, the other had more weight given to certain models based on previous public leaderboard performance. The equally weighted ensemble performed better.

A6. Interesting findings

Most top teams focused on frame-by-frame 2D CNNs. Our approach was primarily based on 3D CNNs.

We devoted little time to “chasing” cross-validation scores. Our models typically had validation losses of around 0.15, which appears higher (less good) than most of the models people have discussed on the Kaggle forums. However, our LB performance appeared much greater. We therefore spent relatively little time optimizing hyperparameters, training schedules etc. locally because we knew such measures correlated poorly with the LB, and instead we focussed on training a variety of solutions with ensembled well with regards to the public LB.

We are confident that use of 3D CNNs helped us greatly: the 3D models we used have similar numbers of parameters to 2D models, and yet also the kernels have a 'depth', so they are relatively modest in their ability to fit in the 2D plane.

We found 3D CNNs did *much* better than other temporal models, such as time-distributed CNNs (CNN feature maps fed into an RNN, e.g. an LSTM), which we and other groups found overfitted with ease.

Interestingly, we found training 3D CNNs using a sequence of uncompressed PNGs (around 500GB worth) rather than MP4s (i.e the same source data, but losslessly compressed) resulted in the 3D CNNs overfitting more: LB scores were worse despite local CV being better. This was likely due to the CNNs finding it easier to overfit to individual actors’ faces, instead of more generalizable artefacts of deepfakes.

All models were trained on videos that were created from the original videos. It was noted that these videos were compressed from the original videos, which may have allowed for better generalizability.

We used CutMix augmentation in several of our models, which may have improved generalizability.

A7. Simple Features and Methods

We did not find that any single model was disproportionately effective, but they synergised particularly well through ensembling. To put the benefits of ensembling with these models into context, when we last tried the I3D model (lowest validation loss) alone we ended up with a public LB score of 0.341. With ensembling the public LB score improved to 0.253. Because of this, it may be unrealistic to expect a single 3D CNN, at least trained on the present dataset, to be able to perform this task.

However, the incremental benefits from each additional 3D CNN in the later stages of the competition was small, and so an ensemble of say 4 3D CNN architectures and a 2D network is likely to be almost as efficacious as our 8 model pipeline.

A8. Model Execution Time

The time taken to train the models varies by architecture, but importantly it is also preceded by pre-processing the face data (reading in a full resolution video file, identifying faces, and extracting smaller videos for analysis): this takes around 1 second per video, and can take several days to perform across the entire DFDC dataset.

One these smaller MP4 files have been extracted, each model up to 24 hours to train on a single RTX Titan GPU, assuming training continues for 20 epochs.

Our submission notebook was able to run inference with the 8 model ensemble on the 400 testing videos in around 45 minutes through the Kaggle notebook interface. This includes loading times and installation of packages. The actual forward pass time per video averaged just under 6 seconds.

A9. References

We used pretrained models from the following sources:

1) Inception 3D, from https://github.com/piergiaj/pytorch-i3d (Apache license)
2) Resnet3D-34 from https://github.com/kenshohara/video-classification-3d-cnn-pytorch (MIT license)
3) MC3, from https://github.com/pytorch/vision/blob/master/torchvision/models/video
4) R2+1D, from https://github.com/pytorch/vision/blob/master/torchvision/models/video
5) SE-ResNeXt50 from https://github.com/Cadene/pretrained-models.pytorch