DFDF – 3rd Place Solution

Summary

- Ensemble of 3 EfficientNet-B7 models
  - 2 “frame-by-frame models”
  - 1 “sequence-based model”
    (with addition of 3D convolution on each EfficientNet-B7 block)

- How to solve the Overfitting problem
  - Mixup technique on aligned real-fake pairs

Summary

- Augmentations
  - AutoAugment
  - Random Erasing
  - Random Crops
  - Random Flips
  - Various video compression parameters (done on-the-fly)
    - Short cropped tracks (50 frames each) → saved in PNG format
    - Loaded & reencoded with random parameters(by ffmpeg)
      at each training iteration

Summary

- Inference stage
  - Due to mixup →model predictions were “uncertain”
  - Used a simple transformation
  - Final prediction
    - Average of the predictions of models with weights proportional to confidence

- Total training and preprocessing time
  - 5 days on DGX-1

Summary

- Hardware architecture
  - CPU: **Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz**
  - GPU: **8x NVIDIA Tesla V100 SXM2 32 GB**
  - RAM: **512 GB**
  - SSD: **6 TB**

Key ingredients

1. Mixup on aligned real-fake pairs
2. Video compression augmentation

Mixup on aligned real-fake pairs

- Severe overfitting problem
  - All models overfitted in 2-3 epochs
    (validation loss increased)

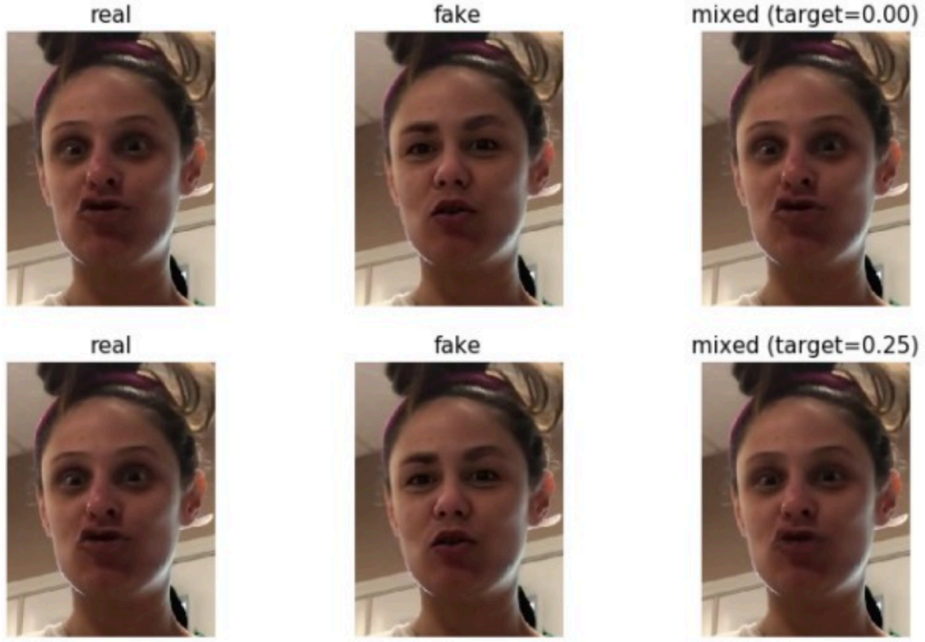
TO SOLVE

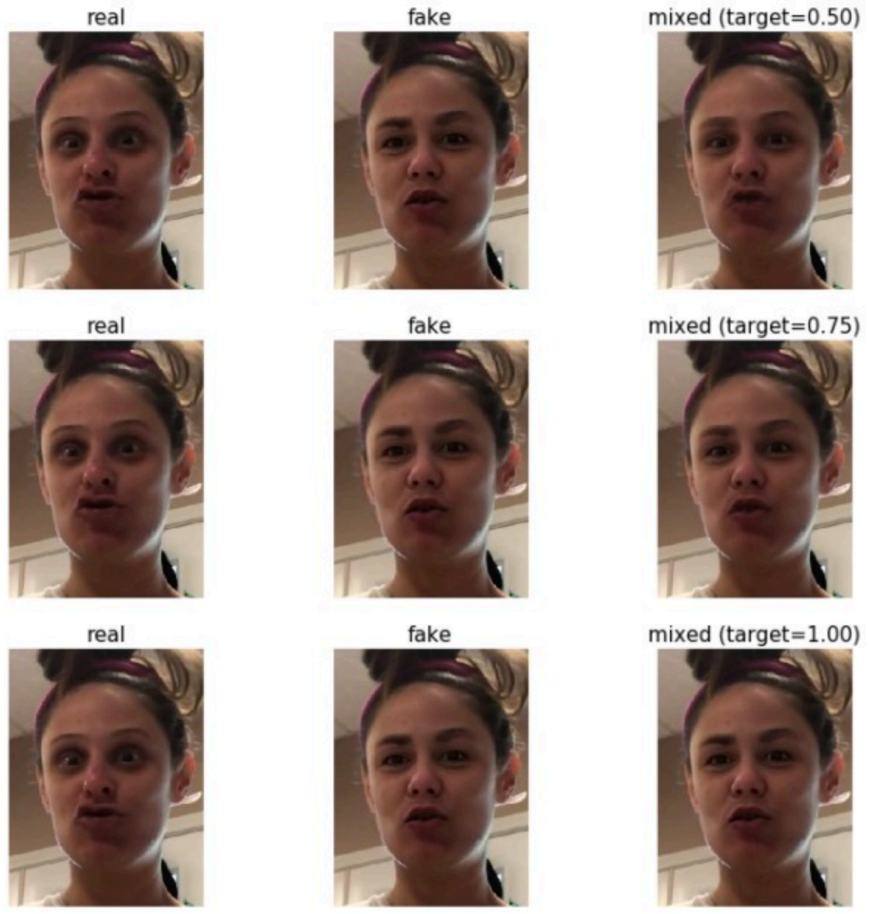
- Mixup on aligned real-fake pairs
  - Take the corresponding real face from the original video
    (same box coordinates & same frame number)
  - Do a linear combination of them(real & fake)

```
input_tensor = (1.0 - target) * real_input_tensor + target * fake_input_tensor
```

Mixup on aligned real-fake pairs

- Real & fake samples are aligned
  → background remains almost unchanged on interpolated samples
  → **reduces overfitting** & **make the model pay more attention to the face**





Video compression augmentation

- Augmentations close to degradations are applied (similar to real-life video)
  - Reduce the FPS of the video to 15
  - Reduce the resolution of the video to ¼ of its original size
  - Reduce the overall encoding quality

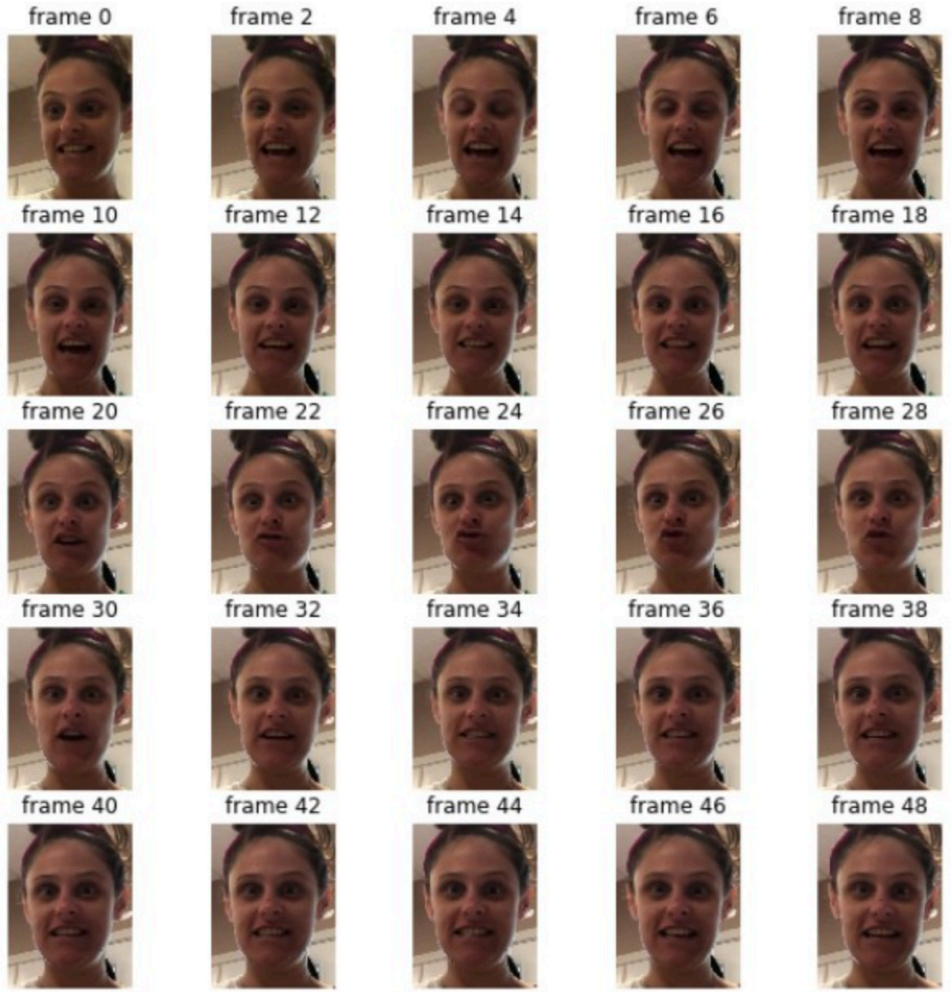
Video compression augmentation

1. Augmentation with random parameters of video encoding
(to make the model resistant to various parameters of video compression)
    - ~~applied to the original video on-the-fly during training~~ (IMPOSSIBLE)
    - applied on the cropped(1.5x areas around the face) short(50 frames) clips
    - Each clip was saved as separate frames in png format

2. On-the-fly augmentation
    - “ffmpeg-python” was used
    - At each iteration, the following parameters were randomly sampled
        - FPS (15 to 30)
        - scale (0.25 to 1.0)
        - CRF (17 to 40)
        - random tuning option

Video compression augmentation

1. Augmentation with random parameters of video encoding



Model architecture

1. Frame-by-frame models **(2 models)**
2. Sequence-based model **(1 model)**

Model architecture

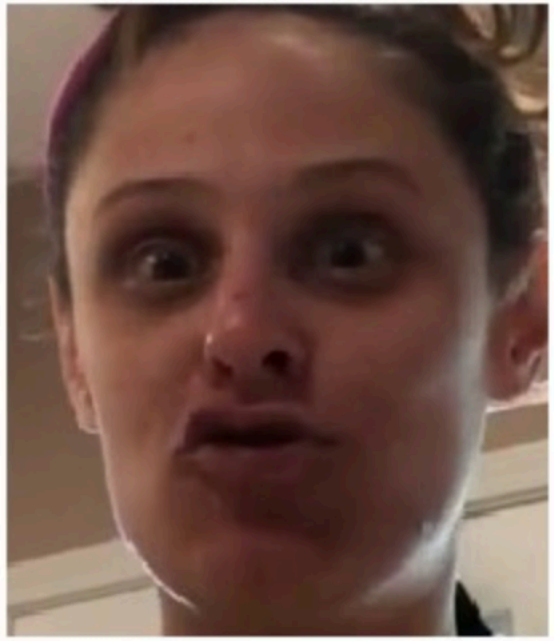
- "**EfficientNet-B7 with Noisy Student pre-trained weights**" was used
- The size of the input image: 224x192
  (most of the faces in the training dataset are smaller)
- The final ensemble consist of **three models**
  - **2** frame-by-frame models
  - **1** sequence-based model

Frame-by-frame models

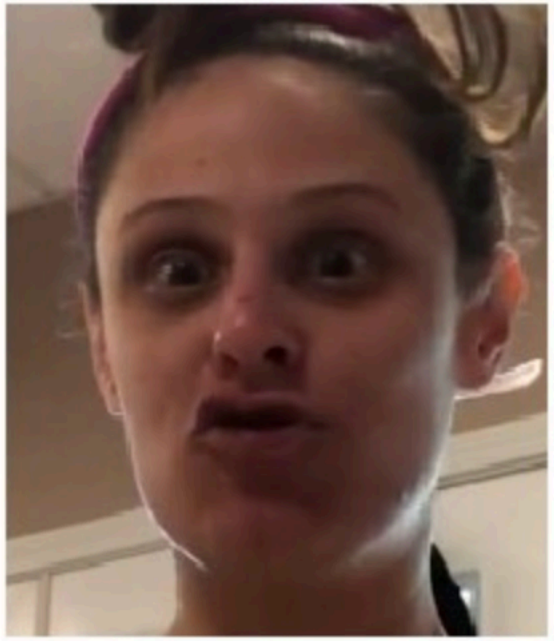
- Each model differ in
  the size of the area around the face & augmentations during training

- Example of input images for each of the models

first model input



second model input



Sequence-based model

- “time dependencies” can be useful for detecting fakes
- Addition of a **3d convolution to each block of the EfficientNet model**
- This model worked **slightly better** than similar frame-by-frame model

- The length of the input sequence: **7 frames**
- The step between frame: **1/15 of a second**

- Example of an input sequence

Image augmentations

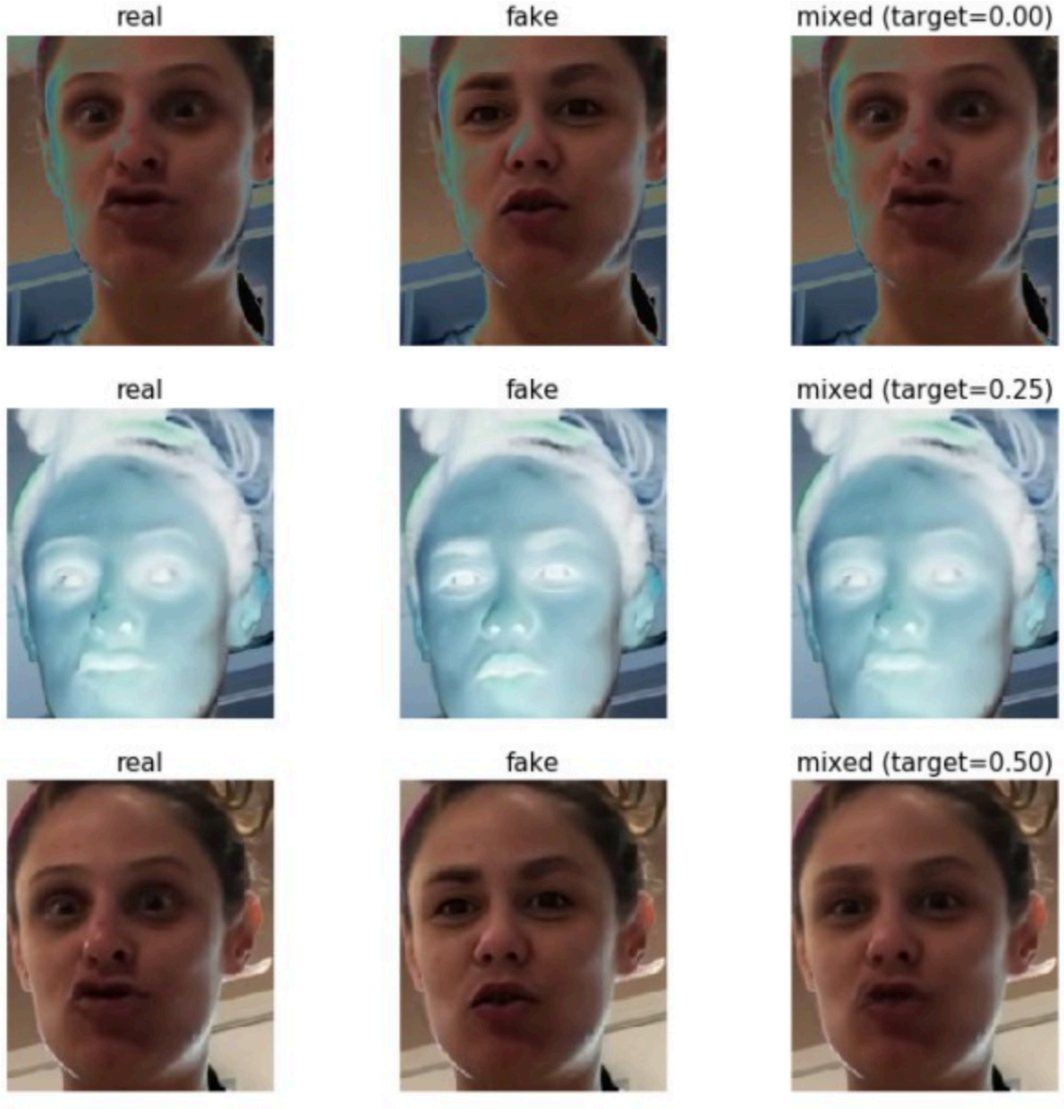
1. AutoAugment
2. Random Erasing
3. Random Crops
4. Random Horizontal Flips

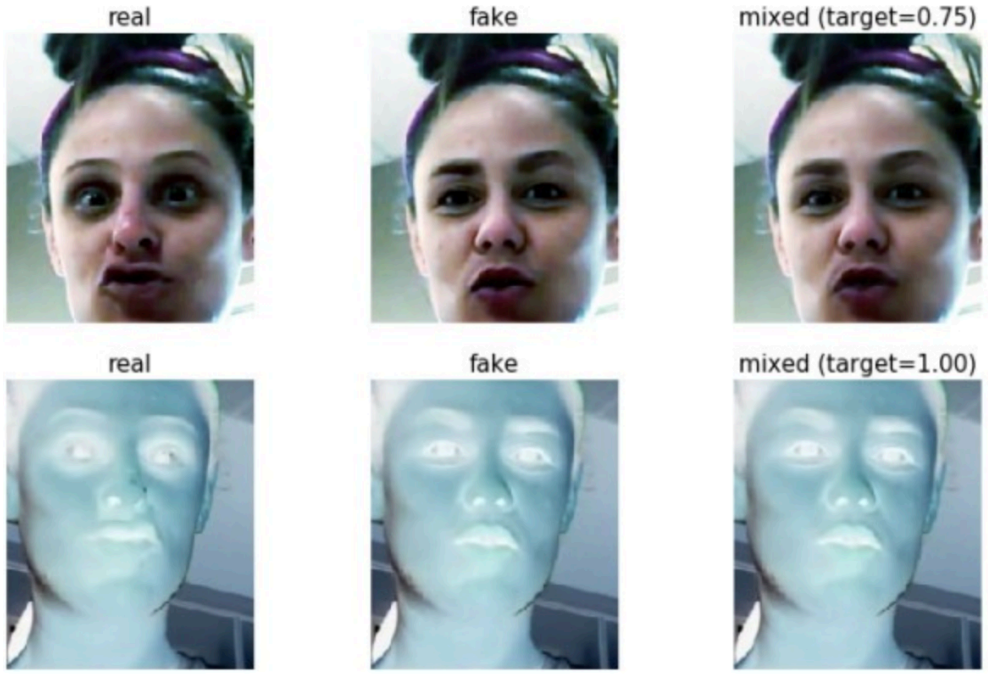
Image augmentation

- We should augment **real-fake pairs** **the same way** (**since we used mixup**)

- We should augment **frames that belong to the same clip** **in the same way** **for a sequence-based model**

Image augmentation

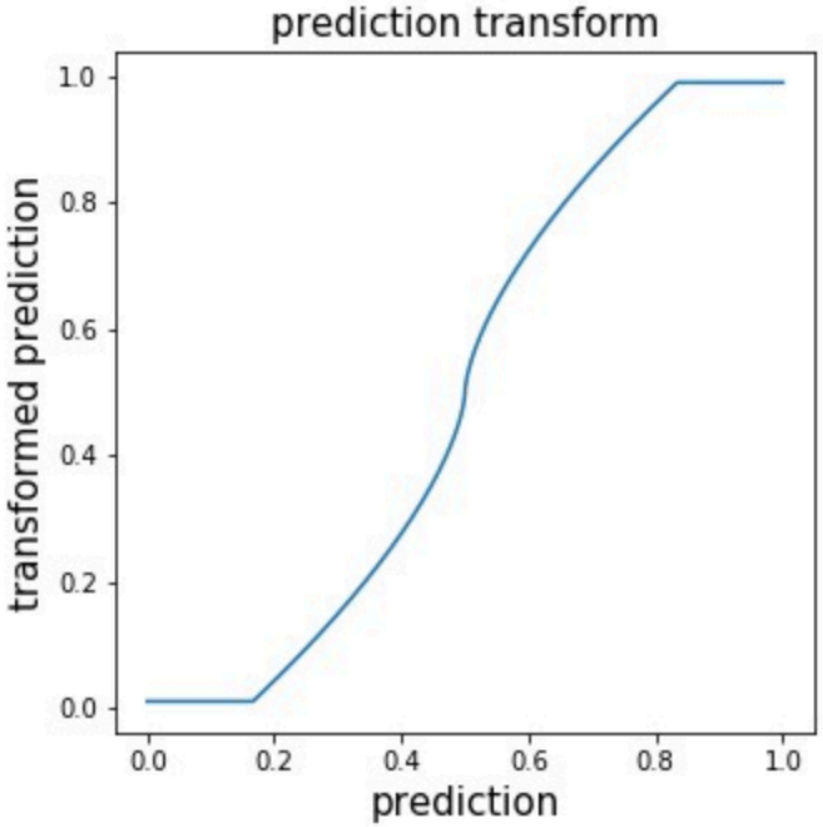




Inference post-processing

Inference post-processing

- **Due to mixup**
  - Predictions of the models were uncertain
  - Not optimal for the logloss
- **To increase confidence**
  - Applied the following transformation



Inference post-processing

- Due to computational limitations
  predictions are made on a subsample of frames
  - Half of the frames were orizontally flipped

- The prediction for the video is obtained by
  averaging all the predictions with weights proportional to the confidence
  (the closer the prediction to 0.5, the lower its weight)
  - Such averaging works like attention
    because the model gives predictions close to 0.5 on poor quality frames (profile faces, blur, etc.)

References

- https://github.com/NTech-Lab/deepfake-detection-challenge#the-hardware-we-used
- Brian Dolhansky, Russ Howes, Ben Pflaum, Nicole Baram, Cristian Canton Ferrer, “The Deepfake Detection Challenge (DFDC) Preview Dataset”
- https://trac.ffmpeg.org/wiki/Encode/H.264
- Qizhe Xie, Minh-Thang Luong, Eduard Hovy, Quoc V. Le, “Self-training with Noisy Student improves ImageNet classification”
- Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le, “AutoAugment: Learning Augmentation Policies from Data”