Winning Model Documentation: Model Summary

Team \WM/, 4th place in Deepfake Detection Challenge

This is the model summary documentation for Team \WM/ in the *Deepfake Detection Challenge Competition*. We will introduce the background of our team and explain the details of our solution in the following pages.

Please also refer to our open-sourced code. The README.md file in it serves as a guideline to reproduce our submission.

Part I: Background on Our Team

In this part, we introduce the background of our team and give responses to the questions listed in the *Winning Model Documentation Guidelines*.

1. Competition Information

- **Competition Name:** Deepfake Detection Challenge
- **Team Name:** \WM/
- **Public Leaderboard Score:** 0.28680
- **Private Leaderboard Score:** 0.42842
- **Private Leaderboard Place:** 4th

2. Team Members' Information

- **Name:** Wenbo Zhou
- **Location:** Hefei, China
- **Email:** welbeckz@ustc.edu.cn

- **Name:** Hao Cui
- **Location:** Hefei, China
- **Email:** cuihao.leo@gmail.com

- **Name:** Hanqing Zhao
- **Location:** Hefei, China
- **Email:** zhq2015@mail.ustc.edu.cn

3. Additional Questions

- **What is your academic/professional background?**

  We are a group of researchers from University of Science and Technology of China (USTC). Our supervisors are Prof. Weiming Zhang and Prof. Nenghai Yu.

Wenbo Zhou is a Postdoctoral researcher. His research interests include Computer Vision and AI Security.

Hao Cui is a master's student. His research interests include Adversarial Examples and Digital Image Watermarking.

Hanqing Zhao is a master's student. His research interests include Image Classification and Information Hiding.

- **Did you have any prior experience that helped you succeed in this competition?**

  Wenbo Zhou and Hao Cui had participated in IJCAI-19 Alibaba Adversarial AI Challenge held by Alibaba Inc and IJCAI 2019 committee. The competition experience helps us succeed in this Deepfake Detection Challenge. This is the first time for Hanqing Zhao to participate in Kaggle competition.

- **What made you decide to enter this competition?**

  Deepfake Detection is a popular topic in recent years, which is also a related topic of our research team. The generous prizes also motivate us to enter the competition.

- **How much time did you spend on the competition?**

  We join the competition at the very beginning and try some simple idea intermittently. We spend most time working on this competition during the last month.

- **If part of a team, how did you decide to team up?**

  We work in same research lab and cooperate in research. Wenbo Zhou is experienced in deepfake related research topic. Hao Cui and Hanqing Zhao have strong coding skills.

- **If you competed as part of a team, who did what?**

  All of the team members contribute a lot for the competition. Specifically, Wenbo Zhou and Hanqing Zhao mainly work on training different detection models. Hao Cui helps promoting the execution efficiency of our codes.

Part II: Method Details

In this part, we first give an overall description of our solution, and we respond to part of the questions listed in Kaggle's *Winning Model Documentation Guidelines* to explain our solution more clearly.

1. Summary

Our final solution ensembles two WS-DAN[1] models (with EfficientNet-b3[2] and Xception[3] feature extractors, respectively) and a Xception classifier to produce per-face predictions. Per-video prediction confidence is calculated by averaging all frames' predictions.

We implement our method with PyTorch deep-learning framework. We extracted and aligned faces in the competition dataset as training data. Each model was trained on 1080Ti/2080Ti GPUs for 20~50 epochs.

2. Methodology Evolution

We have tried multiple different solutions before the current best one. At the beginning, we tried CNN-LSTM with optical flow as input, but the results were very bad. We gave up and switched to per-frame prediction with binary CNN classifier as most other competitors did.

We trained Xception and some other CNN architectures on aligned face images. The log loss is around 0.40 (public leaderboard, LB) and 0.20 (local cross-validation, CV). By introducing a sets of strong augmentation, we further improve log loss to around 0.35~0.38 (LB) and 0.10~0.15 (CV). Further training improves local CV results but not LB results.

We also tried to double the width of Xception, but it became more difficult to converge and did not perform better. Given the low local CV log loss, we assumed that Xception-like architectures are large enough for this binary classification task. So we stopped exploring larger models and did more efforts for mitigating overfitting.

We found WS-DAN (Weakly Supervised Data Augmentation Network)[1] as an enhancement. Our adapted WS-DAN model with Xception feature extractor achieved around 0.30 LB score.

We hoped to improve the score by model ensemble. We started to train another WS-DAN model with EfficientNet-b3 feature extractor, but we didn't have enough time to complete training before the competition deadline. At last, we ensembled a WS-DAN with Xception as feature extractor, a semi-finished WS-DAN with EfficientNet-b3 and a previously trained Xception model. Our final public-LB score is 0.28680.

3. Final Solution

As mentioned, our final solution ensembles two WS-DAN models with different feature extractors and a Xception model to produce per-frame/face prediction with aligned faces as input. Per-video prediction is calculated by averaging all frames' predictions.

The flowchart of our solution is depicted in Figure 1.

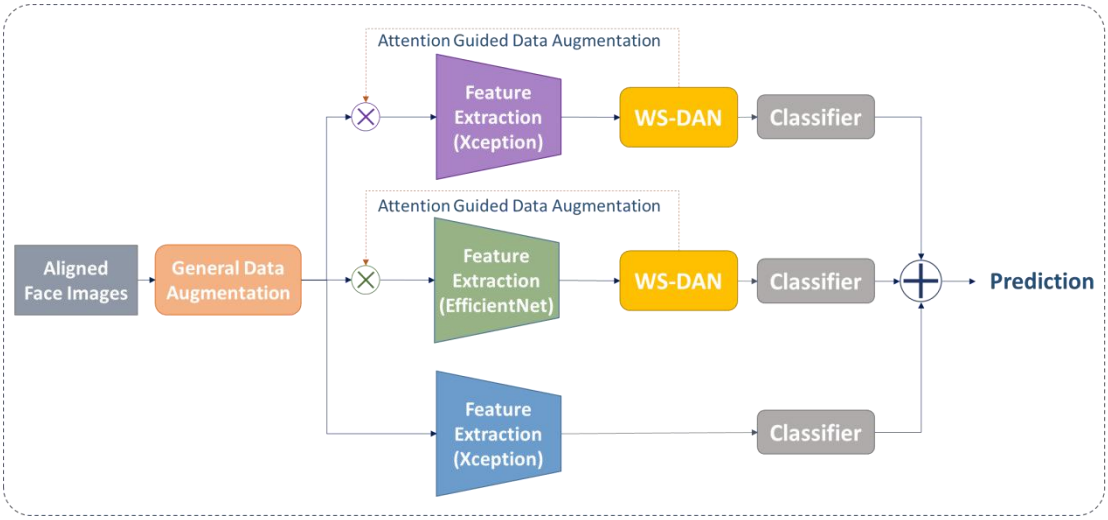

Figure 1: The flowchart of our final solution

Data Preparation

- For each frame of video, we used RetinaFace[4] to extract and produce aligned face images.
- If there are more than one face detected in a frame, only the largest one is extracted.
- All aligned face images are resized to 320x320 and saved.
- We calculated the mean and variance of sampled face images as normalization parameters (in WS-DAN).

Augmentation

During training, we applied following augmentation pipeline:

- Horizontal Flip
- Gaussian Noise / ISO Noise
- Blur, including Motion Blur and Gaussian Blur
- Random Hue-Saturation Modification
- Random Brightness/Contrast Modification
- Image Sharpen or Emboss
- Sepia Filter

For the exact augmentation pipeline we used, please refer to our training code.

We used this augmentation setting just because we had used similar set of augmentations when training Xception model and it had improved results. We had chosen the setting intuitively with no much special consideration. We did not use any external data.

Network Design & Adaption

We had been using Xception model before WS-DAN was found to be effective. Xception generally converge faster than other image classification networks. We just used the original design with two-class output and applied some augmentation to improve model generalization.

After reaching the bottleneck of Xception, we started seeking alternative methods for model generalization. In practice, random data augmentation is usually low-efficiency and might introduce many uncontrolled back-ground noises. We introduce WS-DAN as part of the architecture of our network to enhance the ability of data augmentation, which helps a lot to promote the performance of our models.

For each training image, WS-DAN generates attention maps to represent the object's discriminative parts by weakly supervised learning. Then, WS-DAN augments the image guided by these attention maps, including attention cropping and attention dropping. WS-DAN improves the classification accuracy in two folds: In the 1st stage, images can be seen better since more discriminative parts' features will be extracted. In the 2nd stage, attention regions provide accurate location of object, which ensures our model to look at the object closer and further improve the performance.

We give some samples of images augmented by attention drop in Figure 2.

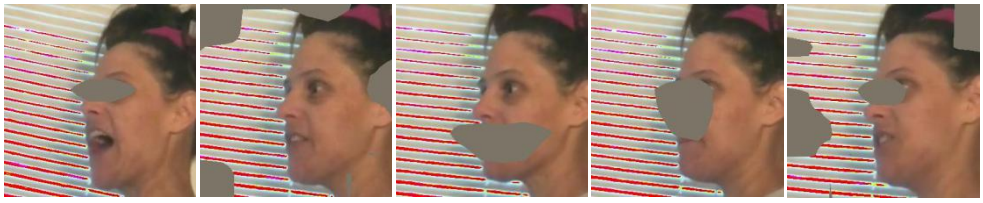

Figure 2: Samples of images augmented by attention drop

For WS-DAN models, we did following adaptations:

- We adjusted the attention number and parameters in loss function for task adaption because the original settings are designed for several fine-grained visual classification tasks.
- We added dropout on feature matrix because we found that there are attention heads that likely appears on the edge of images, it seems meaningless and attention cropping cannot punish it. We think these attentions is cheat for attention dropping. We assumed drop out can force every attention heads focus on meaningful information.
- We multiplied feature centers by 2 after each epoch for gradually fixing feature centers.

We first used WS-DAN with Xception feature extractor which was initialized from our previous trained Xception model. To further improve performance, we seek to ensemble multiple WS-DAN models with different extractors. We found

EfficientNet-b3[2] is a good tradeoff for training time and accuracy considering the limitation of our GPU resources.

4. Additional Questions

- **What were the most important features?/How did you select features?**

  We use data augmentation during the training stage, details are described in the Augmentation part. We tried different augmentations, the selected ones are verified efficient for promoting the performance when making a local test on Xception model.

- **What training methods did you use?/Did you ensemble the models?/How did you weight the different models? (Training Methods)**

  We used WS-DAN as our main training method, using Xception and EfficientNet-b3 as backbone. We adjusted the attention numbers and parameters in loss function of WS-DAN to fit Deepfake detection task. We used dropout, feature center fixing and strong data augmentations to mitigate overfitting. However, due to the limitation of GPU resources, we didn't have enough time to complete the training on EfficientNet-b3 based WS-DAN. The final solution emsembles a WS-DAN with Xception as feature extractor, a semi-finished WS-DAN with EfficientNet-b3 as feature extractor, and a previously trained Xception model.

  The weights are as follows:

  WS-DAN (Xception): 0.7; WS-DAN (EfficientNet-b3):0.1; Xception: 0.2

- **How long does it take to train your model?/How long does it take to generate predictions using your model? (Model Execution Time)**

  On a server with 4~6 1080Ti/2080Ti GPUs, it takes about 1 day to train a good Xception model (20+ epochs) and about one week to train a WS-DAN model (around 50 epochs). And it takes about 40 minutes to generate predictions on 400 test videos locally.

- **What the most important trick you used?/What do you think set you apart from other in the competition? (Interesting findings)**

  We think that WS-DAN contributes a lot in our final solution. The augmented images generated by weakly supervised attention learning help promote the performances of models. It might also enhance the transferability of models on mismatched datasets.

References

[1] Hu, T., Qi, H., Huang, Q., & Lu, Y. (2019). See better before looking closer: Weakly supervised data augmentation network for fine-grained visual classification. *arXiv preprint arXiv:1901.09891*. (Code: https://github.com/GuYuc/WS-DAN.PyTorch)

[2] Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*. (Code: https://github.com/lukemelas/EfficientNet-PyTorch)

[3] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).

[4] Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., & Zafeiriou, S. (2019). Retinaface: Single-stage dense face localisation in the wild. *arXiv preprint arXiv:1905.00641*. (Code: https://github.com/biubug6/Pytorch_Retinaface)