Post-Train Adaptive MobileNet for Fast Anti-Spoofing

Kostiantyn Khabarlak[a]

[a] *Dnipro University of Technology, D. Yavornytskoho Av., 19, Dnipro, 49005, Ukraine*

Abstract

Many applications require high accuracy of neural networks, as well as low latency and user data privacy guaranty. Face anti-spoofing is one of such tasks. However, a single model might not give the best results for different device performance categories, while training multiple models is time consuming. In this work we present Post-Train Adaptive (PTA) block. Such a block is simple in structure and offers a drop-in replacement for MobileNetV2 Inverted Residual block. PTA block has multiple branches with different computation costs. The branch to execute can be selected on-demand and at runtime, thus offering different inference times and configuration capability for multiple device tiers. Crucially, the model is trained once and can be easily reconfigured after training, even directly on a mobile device. In addition, the proposed approach shows substantially better overall performance in comparison to the original MobileNetV2 as tested on CelebA-Spoof dataset. Different PTA block configurations are sampled at training time, which also decreases overall wall-clock time needed to train the model. While we present computational results for the anti-spoofing problem, the MobileNetV2 with PTA blocks is applicable to any problem where the original MobileNetV2 was, which makes the results presented practically significant.

Keywords 1

Neural Network Adaptation, Post-Train Adaptive, Inference Speed, Mobile Computing, Edge Computing, Anti-Spoofing, Computer Vision

1. Introduction

Convolutional neural networks have shown an extraordinary performance in computer vision tasks. While the initial research has been focused purely on quality regardless computation cost, the modern research trend is to design fast yet accurate neural networks, and in significant part such a trend has been motivated by the requirements of low-latency data processing, user data privacy, as well as reduction of server load. In addition to the fact Mobile and IoT devices offer significantly less computational power, typically several generations or price categories of such devices should be considered, yet architecture of most modern neural networks can only be configured before training and not after. This leaves us with two alternatives: 1) to train a separate network for each device category, which requires more time and effort; 2) to design a single architecture which will target either high-end devices and high quality, or compatibility with all device generations at the cost of accuracy. Both of these solutions are suboptimal.

Real-time time face anti-spoofing is one of the algorithms that is preferable to be performed directly on a mobile device. The anti-spoofing task is to distinguish whether the user shows its real, live face, or a recording of someone else's. The problem is complicated by plethora of ways spoofing attack can be performed, such as printed face image, poster, video, face mask, etc. Anti-spoofing can be found as a component in face-based access control systems, where it is not acceptable if access can be granted to an unauthorized person holding someone's photograph.

In this work we propose Post-Train Adaptive (PTA) block, which is simple in structure and offers a drop-in replacement for MobileNetV2 Inverted Residual block. The PTA block has multiple branches

IntelITSIS'2022: 3rd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 23–25, 2022, Khmelnytskyi, Ukraine

EMAIL: habarlack@gmail.com (K. Khabarlak);
ORCID: 0000-0003-4263-0871 (K. Khabarlak);

with different computation costs. The training procedure is constructed in way, so that the branch to infer on in a fully-trained network can be selected on-demand and at runtime, thus offering a way to change network inference speed and to target multiple device tiers. Block configuration choice can be made based on device speed, system load, desired power consumption or target quality. Crucially, the model is trained once and can be easily reconfigured after training, even directly on a mobile device.

To summarize, our main contributions are as follows:

1. We introduce Post-Train Adaptive block for MobileNetV2 network, which is capable of switching between different performance/quality levels after being trained and at runtime directly on a mobile device.
2. We demonstrate superiority of the proposed approach over the original MobileNetV2 network both in terms of quality, as well as inference speed on multiple mobile devices on face anti-spoofing problem. The qualitative metrics are provided on CelebA-Spoof dataset.

2. Literature Overview

The initial very deep convolutional network research has been focused on finding exact configurations for convolution blocks (including kernel size and stride), pooling type and activation functions. Each block of these networks was "plain", i.e. contained a single branch, such as in VGG network [1] with up to 19 layers deep. It has been noticed, that in general deep neural networks have better performance and overall generalization capability; however, it has turned out that building even deeper networks faces a vanishing gradient problem, and the training barely proceeds. In [2] a training experiment has been conducted for plain networks with different depth. The first network contained 20 layers, the second was constructed by adding more layers with the total of 56 layers. It has been expected, that if the newly added layers provide no additional benefit, they could be learned to produce an identity mapping, hence, the deeper network should, in theory, show accuracy no less that the shallower network. Yet the experiment has shown that the accuracy of the deeper network was much worse. To counteract the vanishing gradient problem, the authors of ResNet network [2] suggested using an extra identity connection between groups of blocks. Such connection has been termed as skip or residual connection. Also, they have also introduced a Bottleneck block, that is a group of 3 convolutions with kernel size of 1×1, 3×3, 1×1. To limit required computation, 1×1 convolutions reduce and then restore the number of channels, so that a heavier 3×3 convolution processes smaller input (hence the name "bottleneck"). Skip connection is used in this block, so that the input to the first convolution is added to the result of the whole block.

The authors of a widely used MobileNetV2 [3] architecture improve on the ideas previously proposed in ResNet architecture. They introduce an Inverted Residual Block, which on the contrary has small channel count in inputs and outputs, but more channels inside the block. The authors note that such a design is more memory efficient than that of the original ResNet. To keep the number of computations low, lightweight depthwise convolutions are used inside the block. The network has a configurable width parameter, by changing which it is possible to tune the network's computational complexity. A more detailed description of the inverted bottleneck block is provided in the next section.

The following works have improved in the following directions: in Squeeze-and-Excitation Network (SENet) [4] an attention mechanism has been applied to improve the quality of network prediction. In neural networks, attention is used to selectively gate information flowing through the network, so that only the most important components of the signal flow forward. In MnasNet [5], an approach for an automated neural architecture search for mobile or embedded devices has been proposed. During neural network architecture selection process, the best network was selected based on inference speed on an actual mobile device. MobileNetV3 [6] has also improved on the previous approaches by using network architecture search, attention mechanisms and novel activation function. Large and small configurations have been proposed. Mobile neural network inference is important for face-related processing [7] and many other tasks.

Anti-spoofing is used to enhance camera-based access control systems from unauthorized user access based on someone else's photograph, such systems can also be executed directly on mobile [8]. The above-described networks can also be used for anti-spoofing. In general, anti-spoofing can be performed based on RGB signal from a conventional camera, infrared or depth information from special

hardware. For instance, CASIA-SURF [9] dataset, has video information about all three modalities. Using them together improves overall quality, but depth or infrared information is typically not available and requires special hardware. Therefore, in this work we focus on algorithms that use RGB signal only. By using RGB signal anti-spoofing can still be performed by finding color and shape distortions. This is different from image classification, where the shape (and not distortions) is of more importance. In [10] it was proposed to replace convolution operation with Central Difference Convolution, that better captures color gradients to improve anti-spoofing performance. In [11] AENet network was introduced with ResNet as a backbone. The authors utilize rich annotations of the CelebA-Spoof dataset (presented in the same work) to improve network training. Face attribute information (e.g., smile, sunglasses etc.), photo illumination conditions, as well as depth and reflection information are used to form a single multi-task loss. Depth and reflection information is not inherently present in the dataset; hence, the authors propose to infer it from RGB image using an auxiliary neural network. The extra information is used during training and not required during inference. We follow [11], [12] and also use CelebA-Spoof dataset in this work as it is to the best of our knowledge the largest anti-spoofing dataset to date.

While many of the above-described networks offer a capability of configuration change either through separate small/large configuration or through width parameter. The architecture design should be completed prior to training. No capability to change the network configuration after it has been trained is proposed in these networks. In this work we focus on improving MobileNetV2 architecture as it is one of the most widely used networks on mobile devices.

3. Materials and Methods

The main building block of a MobileNetV2 network is an Inverted Residual Block. This block starts processing input with a 1×1 convolution that expands the number of channels. The expansion is controlled by an expansion factor, the authors propose setting it to 6 for all hidden layers. The convolution is then followed by a Batch Normalization [13] and ReLU6 activation layer. Next, 3×3 Depthwise Convolution is applied, followed again by Batch Normalization and ReLU6. In contrast to the ordinary convolution, depthwise convolution computes an output based on a single input channel to reduce computation required. Finally, a 1×1 convolution with Batch Normalization is applied to shrink the channel count back to the original. The final output is summed element-wise with the input (the abovementioned skip connection). The use of such block has allowed the authors to asymptotically reduce the number of multiply-add operations in the network while retaining good quality. The model also has a width multiplier, by changing which it is possible to adjust overall number of multiply-add operations. However, it is not possible to adjust width of the model after training.

The aforementioned Inverted Residual Blocks are typically repeated with the same number of input and output channels several times. In this work we propose to change the number of inverted residual blocks required for model inference based on user demand and after the model training is complete. For that we introduce a Post-Train Adaptive (PTA) block, whose architecture is depicted on Figure 1. The PTA block has 2 branches: the right (heavy) branch is more computationally expensive and is fully equivalent to a pair of Inverted Residual Blocks; the left (light) branch reduces the computation by executing only a single Inverted Residual Block. The branch to be executed is selected based on user configuration and can be changed dynamically at runtime. It is possible to execute either branch exclusively or both at the same time. If both branches are executed, their outputs are averaged element-wise, so that the feature distribution remains the same. The weights are not shared between any of the blocks. We propose to replace three pairs of Inverted Residual Blocks with the largest number of channels with Post-Train Adaptive (PTA) block schemePTA blocks in MobileNetV2 architecture, as is shown in Table 1.

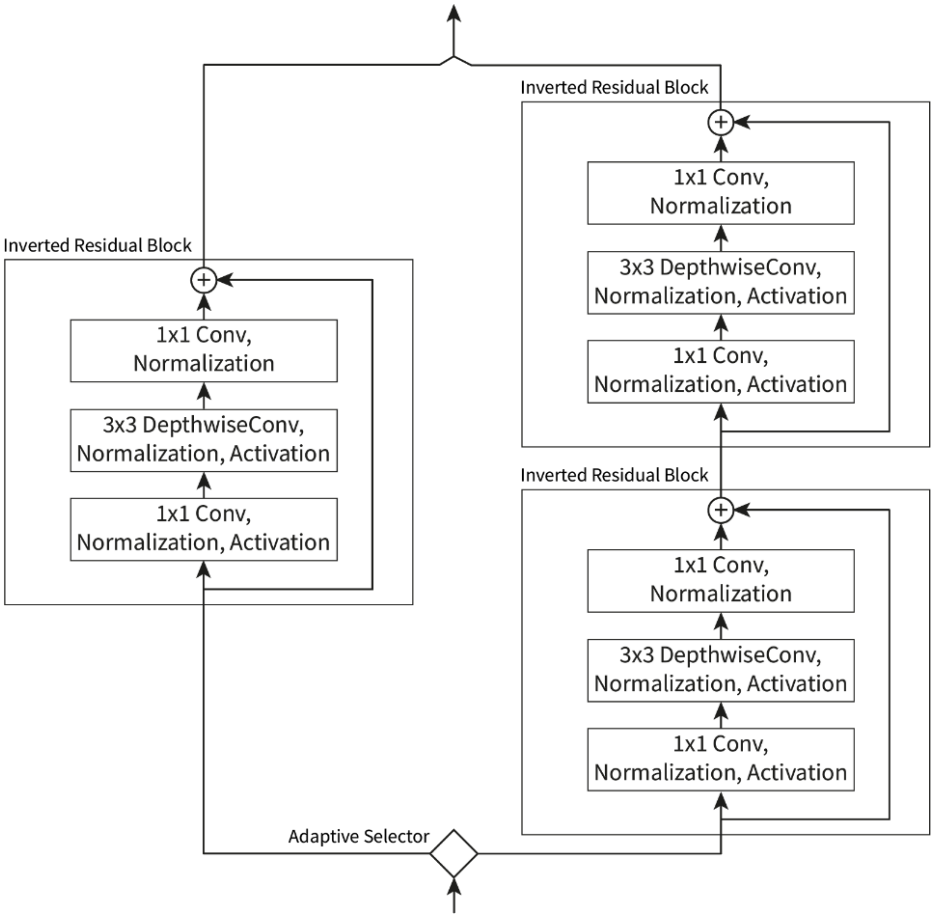

Figure 1: Post-Train Adaptive (PTA) block scheme

Table 1
MobileNetV2 with PTA Architecture

Expansion Ratio	Channels	Repeats	Stride	Block Type
1	16	1	1	Inverted Residual
6	24	2	2	Inverted Residual
6	32	3	2	Inverted Residual
6	64	2	2	Inverted Residual
6	64	1	1	PTA
6	96	1	1	Inverted Residual
6	96	1	1	PTA
6	160	1	2	Inverted Residual
6	160	1	1	PTA
6	320	1	1	Inverted Residual

To train such a model at each iteration we randomly sample a configuration of PTA blocks, and perform forward, then backward pass updating the weights. To avoid excessive randomness in the model, we limit the number of possible configurations for the model to 5: all blocks execute heavy branch; a single of the blocks executes the light branch, while others the heavy one; all of the blocks execute light branch. Configuration sampling is not performed uniformly, we follow the intuition that paths with larger number of weights should be trained for longer and thus assign higher sampling probabilities to such configurations. The exact sampling probabilities are shown in Table 2. Note, that we do not execute both branches at the same time during training. All configurations missing from Table 2 are also assumed to be never sampled and trained on.

Table 2
PTA configurations and corresponding sampling probabilities during training time

PTA Configuration	Sampling Probability
[Heavy, Heavy, Heavy]	0.45
[Light, Heavy, Heavy]	0.15
[Heavy, Light, Heavy]	0.15
[Heavy, Heavy, Light]	0.15
[Light, Light, Light]	0.10
[Both, Both, Both]	0.00

We use Cross Entropy as a loss function. The models that we consider output logits, thus, Cross Entropy also includes softmax computation, and is defined as follows:

$$l(x, y) = -\sum_{n=1}^{N}\sum_{c=1}^{C}\log\frac{\exp(x_{n,c})}{\exp(\sum_{i=1}^{C} x_{n,i})} \quad (1)$$

where N is the number of samples in a mini-batch, $C = 2$ is the number of classes, $x_{n,c}$ is the model logit output for item n and class c. Adam [14] adaptive gradient descent method with learning rate $\alpha = 10^{-4}$ is used as an optimizer.

4. Experiments

To train and evaluate the model we use recent CelebA-Spoof [11] dataset. To the best of our knowledge, this is the largest Anti-Spoofing dataset available to date. Overall, it contains 625,537 pictures (including both spoof and live photos) of 10,177 subjects. Photos are captured with different lighting, environment conditions and different cameras. Only RGB photo information is available in the dataset. Several spoof attack types are considered in the dataset, such as printed full frame photos, paper cut photos, replay attack when picture is presented on a tablet or a phone, and as the authors call it a 3D mask when a printed image is overlayed on top of a human face. In addition to binary spoof/non-spoof label, the dataset contains rich information about spoof type, illumination condition, environment label, as well as face attribute labels (smile, mustache, hat, eyeglasses, etc.). At this point we use only binary spoof/non-spoof information with a possibility of extending our model in future. The dataset defines train/test splits and several evaluation protocols. Our results are given for intra-test protocol, which is used for general model evaluation. We also randomly split the training subset into actually training and validation in 80/20 ratio. Training is performed for 20 epochs. The best model is then selected based on validation set. Gradient computation is performed on mini-batches of size 32 images. The results are reported on the test set.

We also crop images based of face bounding boxes that are present for each image in CelebA-Spoof dataset. The resulting face image is then resized to the resolution of 128×128. We feed color (RGB) images to the model. At training time color jitter and ISO-noise augmentations are used. Note, that no ImageNet pretraining has been used, the models are trained from scratch.

We follow [15], [16] and use the following metrics for model quality evaluation in our paper: Accuracy is a proportion of correctly classified images to the overall number of images; Attack Presentation Classification Error Rate (APCER) is a proportion of attack images incorrectly classified as normal images:

$$APCER = \frac{FP}{FP + TN} \quad (2)$$

Bona Fide Presentation Classification Error Rate (BPCER), that is a proportion of normal (bona) images incorrectly classified as attack images:

$$BPCER = \frac{FN}{FN + TP} \tag{3}$$

Average Classification Error Rate (ACER) is an average of APCER and BPCER:

$$ACER = \frac{APCER + BPCER}{2}, \tag{4}$$

where TP is True Positive, that is the sample is labelled as spoof and the prediction is also spoof, TN is True Negative, meaning both prediction and true label are non-spoof, FP is False Positive, i.e., prediction is spoof, while the image is non-spoof, finally, FN is False Negative, the prediction is non-spoof, but actual image is spoofed.

As in this paper we not only target adaptivity and quality, but also inference time speed on a mobile device, we have selected a pair of Android smartphones for testing. The devices are based on Qualcomm Snapdragon 845 and Snapdragon 800 CPUs, the flagship mobile processors from 2018 and 2013 correspondingly. In terms of modern-day processors, the former can be though-of as mid-to-high-end CPU, and the latter as low-end CPU. These processors are found in many devices; thus, our results can be easily reproduced. Also, in this way we conduct testing on major CPU performance categories.

In addition, we report training time for both MobileNetV2 and MobileNetV2 with PTA blocks on GTX 1050Ti GPU, which is an important metric for practical applications.

5. Results

For the comparison 2 models have been trained: the original MobileNetV2 (hereinafter No PTA) and MobileNetV2 with PTA blocks (hereinafter PTA), constructed as described in Section 3. PTA-based models can be further configured after being training, thus, in all of the following tables we show the configuration for which the testing has been performed. As the proposed MobileNetV2+PTA configuration consists of 3 PTA blocks we use 3-letter abbreviation to denote the exact configuration used. Letter H, L, B are used to denote execution of Heavy, Light, and Both branches correspondingly for each of the PTA blocks.

In Table 3 we show qualitative results as measured on the test set. Accuracy is the higher the better. APCER, BPCER, ACER denote error rates, thus, the lower the better. The best result in each column is shown in red, second best is shown in blue. As can be seen PTA-based models dominate the original MobileNetV2 (No PTA) implementation in all of the metrics. Interestingly, PTA-HHH configuration, which is equivalent in terms of number of parameters and multiply-additions is also better than the original model.

Table 3
Quality comparison with and without PTA blocks

Configuration	Accuracy (↑, %)	APCER (↓, %)	BPCER (↓, %)	ACER (↓, %)
No PTA	96.74	1.07	4.18	2.63
PTA-HHH	96.89	**0.72**	4.12	2.42
PTA-LHH	96.84	**0.70**	4.20	2.45
PTA-HLH	97.04	0.80	3.88	2.34
PTA-HHL	**97.83**	2.30	**2.11**	**2.21**
PTA-LLL	**97.85**	2.53	**1.98**	2.26
PTA-BBB	97.49	1.21	3.05	**2.13**

In Table 4 we present model complexity and inference time comparison. First, we show the number of parameters in each of the models (in millions). For the PTA models we configure the model after training, and then report the number of parameters that is actively used in the corresponding configuration. Next, we measure the number multiply-add operations (in million operations) executed during the forward pass of the model. Also, we measure actual performance on mobile devices on widely popular Snapdragon 845 and Snapdragon 800 processors (hereinafter SD845 and SD800 correspondingly), measured in milliseconds. We have described these processors in more detail in the previous section. Finally, we show relative inference time improvement with respect to the No PTA baseline as measured on SD845. PTA-HHH configuration has the same number of parameters and computation as No PTA and is equivalent in terms of performance on a real device. PTA-BBB configuration uses both Light and Heavy branches in all 3 PTA blocks and thus is slightly more computationally intensive. All other configurations that use a mix of Heavy and Light blocks are faster.

Table 4

Model complexity and inference time comparison with and without PTA blocks

Configuration	# Params (↓, M)	Multiply-Adds (↓, Mops)	Inference Time SD845 (↓, ms)	Inference Time SD800 (↓, ms)	Relative Inference Time (↓)
No PTA	2.23	104.15	21.32	94.23	1.00
PTA-HHH	2.23	104.15	21.27	94.12	1.00
PTA-LHH	2.17	100.63	**18.61**	**82.24**	0.87
PTA-HLH	2.11	**96.50**	19.67	86.96	0.92
PTA-HHL	**1.91**	99.00	19.27	85.15	**0.90**
PTA-LLL	**1.73**	**87.84**	**17.08**	**75.44**	**0.80**
PTA-BBB	2.73	120.47	22.72	100.47	1.07

As we sample Light and Heavy PTA configurations during training, it is expected that overall training time for the PTA-based model should decrease. Our experiments validate this assumption. In Table 5, we demonstrate training time for each of the models joined by the best Accuracy and ACER achieved by each of the models. We show epoch training time in minutes, and overall training time for 20 epochs in hours. Note, the model with PTA blocks is further configured after training, thus, only a single MobileNetV2+PTA has been trained for all the configurations. As can be seen, PTA-based model is better in terms of quality, inference and training time.

Table 5

Qualitative metrics and training time comparison for MobileNetV2 with and without PTA

Configuration	Best Accuracy (↑, %)	Best ACER (↓, %)	Epoch Training Time (↓, m)	Overall Training Time (↓, h)
MobileNetV2	96.74	2.63	49.28	16.43
MobileNetV2+PTA	**97.85**	**2.13**	**43.11**	**14.37**

On Figure 2 we show validation accuracy during training for the original MobileNetV2 (solid blue line) and MobileNetV2+PTA (dashed orange line). For the PTA model we validate on PTA-HHH configuration, which is equivalent in terms of the number of parameters and multiply-adds to the original MobileNetV2. As is clearly seen, the MobileNetV2+PTA has better validation ACER throughout the training process.

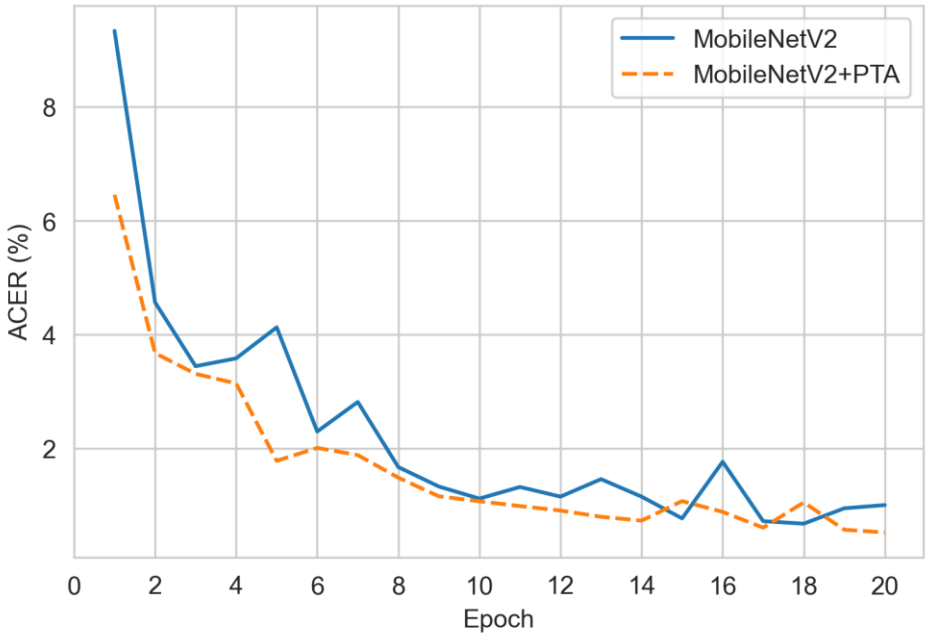

Figure 2: Validation ACER with respect to training epochs

6. Discussion

The key goal of this work is to make it possible to reconfigure a neural network after it has been trained. As has been shown, Post-Train Adaptive block proposed in this work is an efficient way for post-train network configuration. Placing only 3 PTA blocks in MobileNetV2 has made it possible to adaptively adjust inference time from 107% to 80% of the original MobileNetV2 (see Table 4). The simplicity of the PTA block has allowed to implement MobileNetV2 with PTA for inference on mobile devices with different CPUs: high-end Snapdragon 845 and low-end Snapdragon 800. On the latter the inference speed improvement over the original MobileNetV2 is over 18 milliseconds, which is significant for a performance-limited device. The best inference speed is offered by the PTA-LLL model, where all three PTA blocks use light branch only. Interestingly, the second-best inference time is achieved by PTA-LHH configuration with 100.63 Mops and not by PTA-HLH with 96.50 Mops. PTA-LHH is slower than PTA-HLH in 4.72 ms on SD800.

PTA-based configurations have better quality as well. MobileNetV2 (No PTA) and PTA-HHH configurations have the same number of parameters and multiply-additions, but PTA-HHH is better in every metric as seen from Table 3. In this case, the only difference is in training procedure. PTA-based models sample different network configurations during training. Consequently, we suggest, that the proposed training procedure has a positive impact on overall model quality.

Validation ACER comparison depicted on Figure 2 shows that PTA-HHH is better than No PTA model throughout the training procedure. For instance, after a single training epoch these models have achieved ACER of 6.46% and 9.3% for PTA-HHH and No PTA correspondingly, meaning PTA-based model starts to train significantly faster. The final validation ACER for PTA-HHH is 0.53% and is 1.0% for No PTA. We suggest that sampling different block configurations during training makes the network to learn more general features and offers regularization capability. This might explain better PTA-based model results.

Overall, the best accuracy and BPCER is shown by PTA-LLL at 97.85% and 1.98% correspondingly. This is also the fastest configuration. PTA-LHH has the lowest APCER at 0.70% (53% relative performance improvement).

We also investigate a possibility of using multiple branches jointly in PTA-BBB configuration. This is the configuration that shows the best performance in average classification error rate (ACER) at 2.13%, which is a 23.5% relative improvement over the baseline. The model is also better than MobileNetV2 (No PTA) in all other metrics. The 2-branch PTA-BBB model has more parameters

(2.73 M) and multiply-additions (120.47 Mops) that the original MobileNetV2 model with 2.23 M and 104.15 Mops correspondingly. Note, that the PTA-based network is never actually trained with both branches enabled. Therefore, we expect each of the branches to learn slightly different features, thus forming as in-model ensemble similarly to Dropout [17] technique.

The overall training time is lower than that of conventional MobileNetV2 as can be seen from Table 5. On a mainstream Nvidia GTX 1050Ti graphics card we see 2-hour (or 14%) overall training time reduction, when the model is trained for 20 epochs. This significant model training time decrease is achieved by two facts: 1) the heaviest PTA-BBB configuration is never used during training, as is previously mentioned; 2) the actual configurations sampled during training (see Table 2) are lighter than MobileNetV2 (No PTA), which on average results in fewer multiply-add operations performed during both inference and training time. Note, all PTA configurations are obtained from a single trained model. This differs our approach from other found in literature.

7. Conclusion

In this work Post-Train Adaptive block has been first introduced. Such a block is simple in structure and offers a drop-in replacement for a pair of MobileNetV2 Inverted Residual blocks. Thanks to the proposed novel block we improve over MobileNetV2 for anti-spoofing in the following ways: 1) we solve the problem of inability to change the network architecture after it has been trained. The PTA block has light and heavy branches with each of them capable of switching on and off on-demand and at runtime. Not only each of the branches can be used exclusively, but also their prediction can be averaged, forming an in-model ensemble. Therefore, a model can be reconfigured after training to better suit the target device; 2) the lightest PTA configuration shows 20% improvement in terms of actual inference speed on a mobile device, while also having superior quality in comparison to the original MobileNetV2 architecture; 3) the anti-spoofing performance has been substantially improved with PTA-based configurations beating the baseline in all typical anti-spoofing metrics. During training we sample different PTA configurations with different number of parameters. We suggest that this results in the model learning more general features, thus, resulting in better overall quality. All of the aforementioned improvements have been achieved with smaller total training time in comparison to the MobileNetV2 model.

Because-of a significant variation of mobile and edge device computational power, a single neural network targeting several different device categories is suboptimal. The proposed approach, in contrast, allows to train the model once and then adjust its runtime speed according to device characteristics, overall system load and desired battery consumption. This makes the results obtained practically significant.

While the MobileNetV2 with PTA blocks architecture is applicable to any problem, where the original MobileNetV2 was, in this work we have investigated only a single (yet important) practical application, that is mobile face anti-spoofing. In future works we will expand our exploration on other applications and will improve PTA blocks performance and quality even further.

8. Acknowledgments

The work is supported by the state budget scientific research project of Dnipro University of Technology “Development of new mobile information technologies for person identification and object classification in the surrounding environment” (state registration number 0121U109787).

9. References

[1] K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL: http://arxiv.org/abs/1409.1556.

[2] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, in 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.
[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks, 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.
[4] J. Hu, L. Shen, and G. Sun, Squeeze-and-Excitation Networks, 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, 2018, pp. 7132–7141. doi: 10.1109/CVPR.2018.00745.
[5] M. Tan et al., MnasNet: Platform-Aware Neural Architecture Search for Mobile, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, 2019, pp. 2820–2828. doi: 10.1109/CVPR.2019.00293.
[6] A. Howard et al., Searching for MobileNetV3, 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, 2019, pp. 1314–1324. doi: 10.1109/ICCV.2019.00140.
[7] K. Khabarlak and L. Koriashkina, Fast Facial Landmark Detection and Applications: A Survey, Journal of Computer Science and Technology, vol. 22, no. 1, pp. 12–41, Apr. 2022, doi: 10.24215/16666038.22.e02.
[8] K. S. Khabarlak and L. S. Koriashkina, Mobile Access Control System Based on RFID Tags and Facial Information, SACIT, no. 2 (4), pp. 69–74, 2020, doi: 10.20998/2079-0023.2020.02.12.
[9] S. Zhang et al., A Dataset and Benchmark for Large-Scale Multi-Modal Face Anti-Spoofing, IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, 2019, pp. 919–928. doi: 10.1109/CVPR.2019.00101.
[10] Z. Yu et al., Searching Central Difference Convolutional Networks for Face Anti-Spoofing, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, 2020, pp. 5294–5304. doi: 10.1109/CVPR42600.2020.00534.
[11] Y. Zhang et al., CelebA-Spoof: Large-Scale Face Anti-spoofing Dataset with Rich Annotations, Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XII, 2020, vol. 12357, pp. 70–85. doi: 10.1007/978-3-030-58610-2_5.
[12] Y. Zhang et al., CelebA-Spoof Challenge 2020 on Face Anti-Spoofing: Methods and Results, CoRR, vol. abs/2102.12642, 2021, URL: https://arxiv.org/abs/2102.12642.
[13] S. Ioffe and C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, in Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, 2015, vol. 37, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.
[14] D. P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, 2015. URL: http://arxiv.org/abs/1412.6980.
[15] H. Wu, D. Zeng, Y. Hu, H. Shi, and T. Mei, Dual Spoof Disentanglement Generation for Face Anti-spoofing with Depth Uncertainty Learning, CoRR, vol. abs/2112.00568, 2021, URL: https://arxiv.org/abs/2112.00568.
[16] R. Tolosana, M. Gomez-Barrero, C. Busch, and J. Ortega-Garcia, Biometric Presentation Attack Detection: Beyond the Visible Spectrum, IEEE Trans. Inf. Forensics Secur., vol. 15, pp. 1261–1275, 2020, doi: 10.1109/TIFS.2019.2934867.
[17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, CoRR, vol. abs/1207.0580, 2012, URL: http://arxiv.org/abs/1207.0580.