Multi-task Learning For Detecting and Segmenting Manipulated Facial Images and Videos

Huy H. Nguyen[*], Fuming Fang[†], Junichi Yamagishi[*†‡], and Isao Echizen[*†]
[*]SOKENDAI (The Graduate University for Advanced Studies), Kanagawa, Japan
[†]National Institute of Informatics, Tokyo, Japan
[‡]The University of Edinburgh, Edinburgh, UK
Email: {nhhuy, fang, jyamagis, iechizen}@nii.ac.jp

Abstract

Detecting manipulated images and videos is an important topic in digital media forensics. Most detection methods use binary classification to determine the probability of a query being manipulated. Another important topic is locating manipulated regions (i.e., performing segmentation), which are mostly created by three commonly used attacks: removal, copy-move, and splicing. We have designed a convolutional neural network that uses the multi-task learning approach to simultaneously detect manipulated images and videos and locate the manipulated regions for each query. Information gained by performing one task is shared with the other task and thereby enhance the performance of both tasks. A semi-supervised learning approach is used to improve the network's generability. The network includes an encoder and a Y-shaped decoder. Activation of the encoded features is used for the binary classification. The output of one branch of the decoder is used for segmenting the manipulated regions while that of the other branch is used for reconstructing the input, which helps improve overall performance. Experiments using the FaceForensics and FaceForensics++ databases demonstrated the networks effectiveness against facial reenactment attacks and face swapping attacks as well as its ability to deal with the mismatch condition for previously seen attacks. Moreover, fine-tuning using just a small amount of data enables the network to deal with unseen attacks.

1. Introduction

A major concern in digital image forensics is the deepfake phenomenon [1], a worrisome example of the societal threat posed by computer-generated spoofing videos. Anyone who shares video clips or pictures of him or herself on the Internet may become a victim of a spoof-video attack. Several available methods can be used to translate head and

Figure 1. Original video frame (top left), video frame modified using Face2Face method [30] (top right, smooth mask almost completely covers the skin area), using Deepfakes method [1] (bottom left, rectangular mask), and using FaceSwap method [27] (bottom right, polygon-like mask).

facial movements in real time [30, 14] or create videos from photographs [4, 9]. Moreover, thanks to advances in speech synthesis and voice conversion [19], an attacker can also clone a person's voice (only a few minutes of speech are needed) and synchronize it with the visual component to create an audiovisual spoof [29, 9]. These methods may become widely available in the near future, enabling anyone to produce deepfake material.

Several countermeasures have been proposed for the visual domain. Most of them were evaluated using only one or a few databases, including the CGvsPhoto database [25], the Deepfakes databases [2, 16, 17], and the FaceForensics/FaceForensics++ databases [26, 27]. Cozzolino et al. addressed the transferability problem of several state-of-the-art spoofing detectors [11] and developed an autoencoder-like architecture that supports generalization and can be easily adapted to a new domain with simple fine-tuning.

Another major concern in digital image forensics is locating manipulated regions. The shapes of the segmentation

masks for manipulated facial images and videos could reveal hints about the type of manipulation used, as illustrated in Figure 1. Most existing forensic segmentation methods focus on three commonly used means of tampering: removal, copy-move, and splicing [6, 32, 7]. As in other image segmentation tasks, these methods need to process full-scale images. Rahmouni et al. [25] used a sliding window to deal with high-resolution images, as subsequently used by Nguyen et al. [21] and Rossler et al. [26]. This sliding window approach effectively segments manipulated regions in spoofed images [26] created using the Face2Face method [30]. However, these methods need to score many overlapped windows by using a spoofing detection method, which takes a lot of computation power.

We have developed a multi-task learning approach for simultaneously performing classification and segmentation of manipulated facial images. Our autoencoder comprises an encoder and a Y-shaped decoder and is trained in a semi-supervised manner. The activation of the encoded features is used for classification. The output of one branch of the decoder is used for segmentation, and the output of the other branch is used to reconstruct the input data. The information gained from these tasks (classification, segmentation, and reconstruction) is shared among them, thereby improving the overall performance of the network.

2. Related Work

2.1. Generating Manipulated Videos

Creating a photo-realistic digital actor is a dream of many people working in computer graphics. One initial success is the Digital Emily Project [3], in which sophisticated devices were used to capture the appearance of an actress and her motions to synthesize a digital version of her. At that time, this ability was unavailable to attackers, so it was impossible to create a digital version of a victim. This changed in 2016 when Thies et al. demonstrated facial reenactment in real time [30]. Subsequent work led to the ability to translate head poses [14] with simple requirements that are met by any normal person. The Xpression mobile app[1] providing the same function was subsequently released. Instead of using RGB videos as was done in previous work [30, 14], Averbuch et al. and Chung et al. used ID-type photos [4, 9], which are easily obtained on social networks. Combining this capability with speech synthesis or voice conversion techniques [19], attackers are now able to make spoof videos with voices [29, 9], which are more convincingly authentic.

2.2. Detecting Manipulated Images and Videos

Several countermeasures have been introduced for detecting manipulated videos. A typical approach is to treat a video as a sequence of image frames and work on the images as input. The noise-based method proposed by Fridrich and Kodovsky [12] is considered one of the best handcrafted detectors. Its improved version using a convolutional neural network (CNN) [10] demonstrated the effectiveness of using automatic feature extraction for detection. Among deep learning approaches to detection, fine-tuning and transfer learning take advantage of high-performing pre-trained models [24, 26]. Using part of a pre-trained CNN as the feature extractor is an effective way to improve the performance of a CNN [21, 22]. Other approaches to detection include using a constrained convolutional layer [8], using a statistical pooling layer [25], using a two-stream network [31], using a lightweight CNN network [2], and using two cascaded convolutional layers at the bottom of a CNN [23]. Cozzolino et al. created a benchmark for determining the transferability of state-of-the-art detectors for use in detecting unseen attacks [11]. They also proposed an autoencoder-like architecture with which adaptation ability was greatly increased. Li et al. proposed using a temporal approach and developed a network for detecting eye blinking, which is not well reproduced in fake videos [17]. Our proposed method, besides performing classification, provides segmentation maps of manipulated areas. This additional information could be used as a reference for judging the authenticity of images and videos, especially when the classification task fails to detect spoofed inputs.

2.3. Locating Manipulated Regions in Images

There are two commonly used approaches to locating manipulated regions in images: segmenting the entire input image and repeatedly performing binary classification using a sliding window. The segmentation approach is commonly used to detect removal, copy-move, and splicing attacks [6, 7]. Semantic segmentation methods [18, 5] can also be used for forgery segmentation [7]. A slightly different segmentation approach is to return the boxes that represent the boundaries of the manipulated regions instead of returning segmentation masks [32]. The sliding window approach is used more for detecting spoofing regions generated by a computer to create spoof images or videos from bona fide ones [25, 21, 26]. In this approach, binary classifiers for classifying images as spoof or bona fide are called at each position of the sliding window. The stride of the sliding window may equal the length of the window (non-overlapped) [25] or be less than the length (overlapped) [21, 26]). Our proposed method takes the first approach but with one major difference: only the facial areas are considered instead of the entire image. This overcomes the computation expense problem when dealing with large inputs.
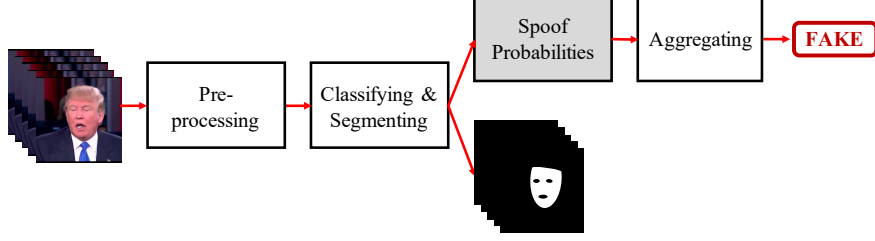
---

[1]https://xpression.jp/

Figure 2. Overview of proposed network.

3. Proposed Method

3.1. Overview

Unlike other single-target methods [22, 11, 7], our proposed method outputs both the probability of an input being spoofed and segmentation maps of the manipulated regions in each frame of the input, as diagrammed in Figure 2. Video inputs are treated as a set of frames. We focused on facial images in this work, so the face areas are extracted in the pre-processing phase. In theory, the proposed method can deal with various sizes of input images. However, to maintain simplicity in training, we resize cropped images to 256×256 pixels before feeding them into the autoencoder. The autoencoder outputs the reconstructed version of the input image (which is used only in training), the probability of the input image having been spoofed, and the segmentation map corresponding to this input image. For video inputs, we average the probabilities of all frames before drawing a conclusion on the probability of the input being real or fake.

3.2. Y-shaped Autoencoder

The partitioning of the latent features (motivated by Cozzolino et al.'s work [11]) and the Y-shaped design of the decoder enables the autoencoder to share valuable information between the classification, segmentation, and reconstruction tasks and thereby improve overall performance by reducing loss. There are three types of loss: activation loss $\mathcal{L}_{act}$, segmentation loss $\mathcal{L}_{seg}$, and reconstruction loss $\mathcal{L}_{rec}$.

Given label $y_i \in \{0, 1\}$, activation loss measures the accuracy of partitioning in the latent space on the basis of the activation of the two halves of the encoded features:

$$\mathcal{L}_{act} = \frac{1}{N} \sum_i |a_{i,1} - y_i| + |a_{i,0} - (1 - y_i)|, \quad (1)$$

where N is the number of samples, $a_{i,0}$ and $a_{i,1}$ are the activation values and defined as the L_1 norms of the corresponding halves of the latent features, $h_{i,0}$ and $h_{i,1}$ (given K is the number of features of $\{h_{i,0}|h_{i,1}\}$):

$$a_{i,c} = \frac{1}{2K} \|h_{i,c}\|_1, \quad c \in \{0, 1\}. \quad (2)$$

This ensures that, given an input x_i of class c, the corresponding half of the latent features $h_{i,c}$ is activated ($a_{i,c} >$
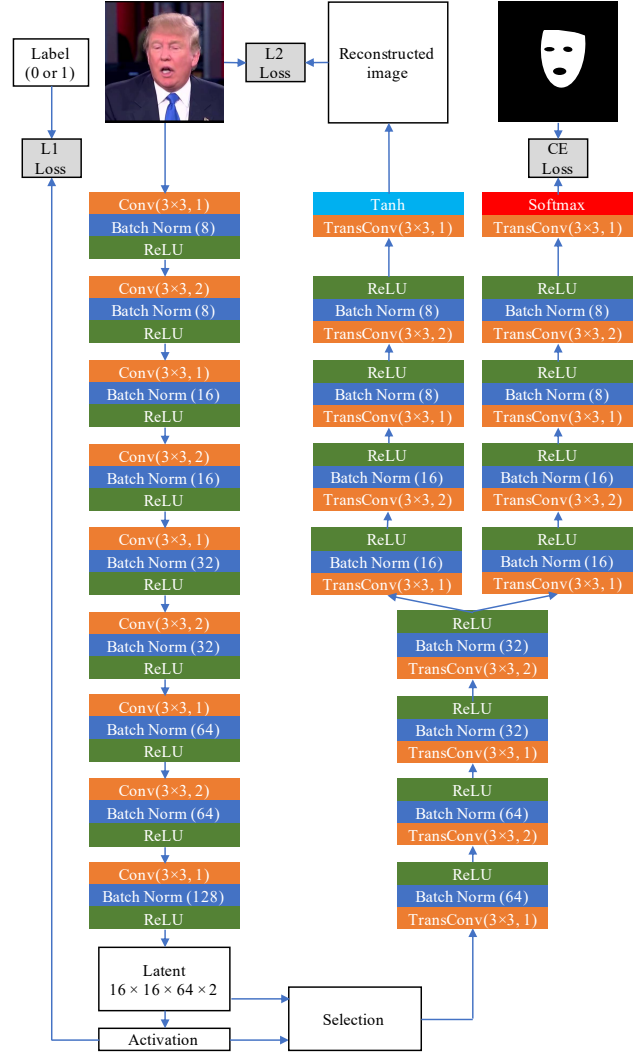

Figure 3. Proposed autoencoder with Y-shaped decoder for detecting and segmenting manipulated facial images.

0). The other half, $h_{i,1-c}$, remains quiesced ($a_{i,1-c} = 0$). To force the two decoders, D_{seg} and D_{rec}, to learn the right decoding schemes, we set the off-class part to zero before feeding it to the decoders ($a_{i,1-c} := 0$).

We utilize cross-entropy loss as the segmentation loss to measure the agreement between the segmentation mask ($s_i = \mathcal{D}_{seg}(\{h_{i,0}|h_{i,1}\})$) and the ground-truth mask (m_i) corresponding to input x_i:

$$\mathcal{L}_{seg} = \frac{1}{N} \sum_i \|m_i \log(s_i) + (1 - m_i) \log(1 - s_i)\|_1. \quad (3)$$

The reconstruction loss uses the L_2 distance to measure the difference between the reconstructed image ($\hat{x} = \mathcal{D}_{rec}(\{h_{i,0}|h_{i,1}\})$) and the original one ($x_i$). For N samples, the reconstruction loss is

$$\mathcal{L}_{rec} = \frac{1}{N} \sum_i \|x_i - \hat{x}_i\|_2. \quad (4)$$

The total loss is the weighted sum of the three activation losses:

$$\mathcal{L} = \gamma_{act}\mathcal{L}_{act} + \gamma_{seg}\mathcal{L}_{seg} + \gamma_{rec}\mathcal{L}_{rec}. \quad (5)$$

Unlike Cozzolino et al. [11], we set the three weights equal to each other (equal to 1). This is because the classification task and the segmentation task are equally important, and the reconstruction task plays an important role in the segmentation task. We experimentally compared the effects of the different settings (described below).

3.3. Implementation

The Y-shaped autoencoder was implemented as shown in Figure 3. It is a fully connected CNN using 3×3 convolutional windows (for the encoder) and 3×3 deconvolutional windows (for the decoder) with a stride of 1 interspersed with a stride of 2. Following each convolutional layer is a batch normalization layer [13] and a rectified linear unit (ReLU) [20]. The selection block allows only the true half of the latent features (h_{i,y_i}) to pass by and zeros out the other half ($h_{i,1-y_i}$). Therefore, the decoders ($\mathcal{D}_{seg}, \mathcal{D}_{rec}$) are forced to decode only the true half of the latent features. The dimension of the embedding is 128, which has been shown to be optimal [11]. For the segmentation branch ($\mathcal{D}_{seg}$), a softmax activation function at the end is used to output segmentation maps. For the reconstruction branch ($\mathcal{D}_{rec}$), a hyperbolic tangent function (tanh) is used to shape the output into the range $[-1, 1]$. For simplicity, we directly feed normalized images into the autoencoder without converting them into residual images [11]. Further work will focus on investigating the benefits of using residual images in the classification and segmentation tasks.

Following Cozzolino et al.'s work [11], we trained the network using the ADAM optimizer [15] with a learning rate of 0.001, a batch size of 64, betas of 0.9 and 0.999, and epsilon equal to 10^{-8}.

4. Experiments

4.1. Databases

We evaluated our proposed network using two databases: FaceForensics [26] and FaceForensics++ [27]. The FaceForensics database contains 1004 real videos collected from YouTube and their corresponding manipulated versions, which are divided into two sub-datasets:

- Source-to-Target Reenactment dataset containing 1004 fake videos created using the Face2Face method [30]; in each input pair for reenactment, the source video (the attacker) and the target video (the victim) are different.

- Self-Reenactment dataset containing another 1004 fake videos created again using the Face2Face method; in each input pair for reenactment, the source and target videos are the same. Although this dataset is not meaningful from the attacker's perspective, it does present a more challenging benchmark than does the Source-to-Target Reenactment dataset.

Each dataset was split into 704 videos for training, 150 for validation, and 150 for testing. The database also provided segmentation masks corresponding to manipulated videos. Three levels of compression based on the H.264 codec[2] were used: no compression, light compression (quantization = 23), and strong compression (quantization = 40).

The FaceForensics++ database is an enhanced version of the FaceForensics database and includes the Face2Face dataset plus the FaceSwap[3] dataset (graphics-based manipulation) and the DeepFakes[4] dataset (deep-learning-based manipulation) [27]. It contains 1,000 real videos and 3,000 manipulated videos (1,000 in each dataset). Each dataset was split into 720 videos for training, 140 for validation, and 140 for testing. The same three levels of compression based on the H.264 codec were used with the same quantization values.

For simplicity, we used only videos with light compression (quantization = 23). Images were extracted from videos using Cozzolino et al.'s settings [11]: 200 frames of each training video were used for training, and 10 frames of each validation and testing video were used for validation and testing, respectively. There is no detailed description of the rules for frame selection, so we selected the first (200 or 10) frames of each video and cropped the facial areas. For all databases, we applied normalization with mean $= (0.485, 0.456, 0.406)$ and standard deviation $= (0.229, 0.224, 0.225)$; these values have been widely used

---

[2] http://www.h264encoder.com/
[3] https://github.com/MarekKowalski/FaceSwap/
[4] https://github.com/deepfakes/faceswap/

Table 1. Design of training and testing datasets.

Name	Source dataset	Description	Manipulation Method	Number of Videos
Training	FaceForensics **Source-to-Target**	Training set used for all tests	Face2Face	704 ×2
Test 1	FaceForensics **Source-to-Target**	Match condition for seen attack	Face2Face	150 ×2
Test 2	FaceForensics **Self-Reenactment**	Mismatch condition for seen attack	Face2Face	150 ×2
Test 3	FaceForensics++ **Deepfakes**	Unseen attack (deep-learning-based)	Deepfakes	140 ×2
Test 4	FaceForensics++ **FaceSwap**	Unseen attack (computer-graphics-based)	FaceSwap	140 ×2

Table 2. Settings for autoencoder.

No.	Method	Depth	Seg. weight	Rec. weight	Rec. loss	Comments
1	*FT_Res*	Shallower	0.1	0.1	L1	Re-implementation of ForensicsTransfer [11] using residual images as input
2	*FT*	Shallower	0.1	0.1	L1	Re-implementation of ForensicsTransfer [11] using normal images as input
3	*Deeper_FT*	Deeper	0.1	0.1	L1	Proposed deeper version of *FT* (*Proposed_Old* method without segmentation branch)
4	*Proposed_Old*	Deeper	0.1	0.1	L1	Proposed method using ForensicsTransfer settings
5	*No_Recon*	Deeper	1	1	L2	Proposed method without reconstruction branch
6	***Proposed_New***	Deeper	1	1	L2	Complete proposed method with new settings

in the ImageNet Large Scale Visual Recognition Challenge [28]. We did not apply any data augmentation to the trained datasets.

The training and testing datasets were designed as shown in Table 1. For the Training, Test 1, and Test 2 datasets, the Face2Face method [26] was used to create manipulated videos. Images in Test 2 were harder to detect than those in Test 1 since the source and target videos used for reenactment were the same, meaning that the reenacted video frames had better quality. Therefore, we call Test 1 and Test 2 the **match** and **mismatch** conditions for a seen attack. Test 3 used the Deepfake attack method while Test 4 used the FaceSwap attack method, presented in the Face-Forensics++ database [27]. These both attack methods were not used to create the training set, therefore they were considered as unseen attacks. For the classification task, we calculated the accuracy and equal error rate (EER) of each method. For the segmentation task, we used pixel-wise accuracy between ground-truth masks and segmentation masks. The *FT_Res*, *FT*, and *Deeper_FT* method could not perform the segmentation task. All the results were at the image level.

4.2. Training Y-shaped Autoencoder

To evaluate the contributions of each component in the Y-shaped autoencoder, we designed the settings as shown in Table 2. The *FT_Res* and *FT* methods are re-implementations of Cozzolino et al.'s method with and without using residual images [11]. They can also be understood as the Y-shaped autoencoder without the segmentation branch. The *Deeper_FT* method is a deeper version of *FT*, which has the same depth as the proposed method. The *Proposed_Old* method is the proposed method using weighting settings from Cozzolino et al.'s work [11], the *No_Recon* method is the version of the proposed method without the reconstruction branch, and the *Proposed_New* method is the complete proposed method with the Y-shaped autoencoder using equal losses for the three tasks and the mean squared error for reconstruction loss.

Since shallower networks take longer to converge than deeper ones, we trained the shallower ones with 100 epochs and the deeper ones with 50 epochs. For each method, the training stage with the highest accuracy for the classification task and a reasonable segmentation loss (if available) was used to perform all the tests described in this section.

4.3. Dealing with Seen Attacks

The results for the match and mismatch conditions for seen attacks are respectively shown in Tables 3 (Test 1) and 4 (Test 2). The deeper networks (the last four) had substantially better classification performance than the shallower ones (the first two) proposed by Cozzolino et al. [11]. Among the four deeper networks, there were no substantial differences in their performances on the classification task. For the segmentation task, the *No_Recon* and *Proposed_New* methods, which used the new weighting settings, had higher accuracy than the *Proposed_Old* method, which used the old weighting settings.

Table 3. Results for Test 1 (image level).

| Method | Classification | | Segmentation |
	Acc (%)	EER (%)	Acc (%)
FT_Res	82.30	14.53	–
FT	88.43	11.60	–
Deeper_FT	**93.63**	7.20	–
Proposed_Old	92.60	7.40	81.40
No_Recon	93.40	**7.07**	89.21
Proposed_New	92.77	8.18	**90.27**

Table 4. Results for Test 2 (image level).

| Method | Classification | | Segmentation |
	Acc (%)	EER (%)	Acc (%)
FT_Res	82.33	15.07	–
FT	87.33	12.03	–
Deeper_FT	92.70	**7.80**	–
Proposed_Old	91.83	8.53	81.40
No_Recon	**92.83**	8.29	89.10
Proposed_New	92.50	8.07	**90.20**

The performances of all methods was slightly degraded when dealing with the mismatch condition for seen attacks. The *FT_Res* and *Proposed_New* methods had the best adaptation ability, as indicated by the lower degradation in their scores. This indicates the importance of using residual images (for the *FT_Res* method) and of using the reconstruction branch (for the Y-shaped autoencoder with new weighting settings: *Proposed_New* method). The reconstruction branch also helped the *Proposed_New* method achieve the highest score on the segmentation task.

4.4. Dealing with Unseen Attacks

4.4.1 Evaluation using pre-trained model

When encountering unseen attacks, all six methods had substantially lower accuracies and higher EERs, a shown in Tables 5 (Test 3) and 6 (Test 4). In Test 3, the shallower methods had better adaptation ability, especially the *FT_Res* method, which uses residual images. The deeper methods,

which had a greater chance of being over-fitted, had nearly random classification results. In Test 4, although all methods suffered from nearly random classification accuracies, their better EERs indicated that the decision thresholds had been moved.

A particularly interesting finding was in the segmentation results. Although degraded, the segmentation accuracies were still high, especially in Test 4, in which FaceSwap copied the facial area from the source faces to the target ones using a computer-graphics method. When dealing with unseen attacks, this segmentation information could thus be an important clue in addition to the classification results for judging the authenticity of the queried images and videos.

Table 5. Results for Test 3 (image level).

| Method | Classification | | Segmentation |
	Acc (%)	EER (%)	Acc (%)
FT_Res	**64.75**	**30.71**	–
FT	62.61	37.43	–
Deeper_FT	51.21	42.71	–
Proposed_Old	53.75	42.00	70.18
No_Recon	51.96	42.45	**70.43**
Proposed_New	52.32	42.24	70.37

Table 6. Results for Test 4 without fine-tuning (image level).

| Method | Classification | | Segmentation |
	Acc (%)	EER (%)	Acc (%)
FT_Res	53.50	43.10	–
FT	52.29	41.79	–
Deeper_FT	53.39	37.00	–
Proposed_Old	**56.82**	36.29	84.23
No_Recon	54.86	35.86	**84.86**
Proposed_New	54.07	**34.04**	84.67

4.4.2 Fine-tuning using small amount of data

We used the validation set (a small set normally used for selecting hyper-parameters in training that differs from the test set) of the FaceForensics++ - FaceSwap dataset [27] for fine-tuning all the methods. To ensure that the amount of data was small, we used only ten frames for each video. We divided the dataset into two parts: 100 videos of each class for training and 40 of each class for evaluation. We trained them using 50 epochs and selected the best models on the basis of their performance on the evaluation set.

The results after fine-tuning for Test 4 are shown in Table 7. Their classification and segmentation accuracies increased around 25% and 8%, respectively, which are remarkable compared with the small amount of data used. The one exception was the *Proposed_Old* method – its segmentation accuracy did not improve. The *FT_Res* method

Table 7. Results for Test 4 after fine-tuning (image level).

Method	Classification		Segmentation
	Acc (%)	EER (%)	Acc (%)
FT_Res	80.04 (↑ 26.54)	17.57 (↓ **25.53**)	–
FT	70.89 (↑ 18.60)	25.56 (↓ 16.23)	–
Deeper_FT	82.00 (↑ 28.61)	17.33 (↓ 19.67)	–
Proposed_Old	78.57 (↑ 21.75)	20.79 (↓ 15.50)	84.39 (↑ 0.16)
No_Recon	82.93 (↑ 28.07)	16.93 (↓ 18.93)	92.60 (↑ 7.74)
Proposed_New	**83.71 (↑ 29.64)**	**15.07 (↓ 18.97)**	**93.01 (↑ 8.34)**

had better adaptation than the *FT* one, which supports Cozzolino et al.'s claim [11]. The *Proposed_New* method had the highest transferability against unseen attacks as evidenced by the results in Table 7.

5. Conclusion

The proposed convolutional neural network with a Y-shaped autoencoder demonstrated its effectiveness for both classification and segmentation tasks without using a sliding window, as is commonly used by classifiers. Information sharing among the classification, segmentation, and reconstruction tasks improved the network's overall performance, especially for the mismatch condition for seen attacks. Moreover, the autoencoder can quickly adapt to deal with unseen attacks by using only a few samples for fine-tuning. Future work will mainly focus on investigating the effect of using residual images [11] on the autoencoder's performance, processing high-resolution images without resizing, improving its ability to deal with unseen attacks, and extending it to the audiovisual domain.

Acknowledgement

References

[1] Terrifying high-tech porn: Creepy 'deepfake' videos are on the rise. https://www.foxnews.com/tech/terrifying-high-tech-porn-creepy-deepfake-videos-are-on-the-rise. Accessed: 2019-03-12. 1

[2] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen. MesoNet: a compact facial video forgery detection network. In *WIFS*. IEEE, 2018. 1, 2

[3] O. Alexander, M. Rogers, W. Lambeth, J.-Y. Chiang, W.-C. Ma, C.-C. Wang, and P. Debevec. The digital emily project: Achieving a photorealistic digital actor. *IEEE Computer Graphics and Applications*, 30(4):20–31, 2010. 2

[4] H. Averbuch-Elor, D. Cohen-Or, J. Kopf, and M. F. Cohen. Bringing portraits to life. *ACM Transactions on Graphics*, 2017. 1, 2

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. 2

[6] J. H. Bappy, A. K. Roy-Chowdhury, J. Bunk, L. Nataraj, and B. Manjunath. Exploiting spatial structure for localizing manipulated image regions. In *ICCV*, pages 4970–4979, 2017. 2

[7] J. H. Bappy, C. Simons, L. Nataraj, B. Manjunath, and A. K. Roy-Chowdhury. Hybrid lstm and encoder-decoder architecture for detection of image forgeries. *IEEE Transactions on Image Processing*, 2019. 2, 3

[8] B. Bayar and M. C. Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *IH&MMSEC*. ACM, 2016. 2

[9] J. S. Chung, A. Jamaludin, and A. Zisserman. You said that? In *BMVC*, 2017. 1, 2

[10] D. Cozzolino, G. Poggi, and L. Verdoliva. Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. In *IH&MMSec*. ACM, 2017. 2

[11] D. Cozzolino, J. Thies, A. Rössler, C. Riess, M. Nießner, and L. Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510*, 2018. 1, 2, 3, 4, 5, 6, 7

[12] J. Fridrich and J. Kodovsky. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 2012. 2

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015. 4

[14] H. Kim, P. Garrido, A. Tewari, W. Xu, J. Thies, M. Nießner, P. Pérez, C. Richardt, M. Zollhöfer, and C. Theobalt. Deep video portraits. In *SIGGRAPH*. ACM, 2018. 1, 2

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICML*, 2015. 4

[16] P. Korshunov and S. Marcel. Deepfakes: a new threat to face recognition? assessment and detection. Idiap-RR Idiap-RR-18-2018, Idiap, 2018. 1

[17] Y. Li, M.-C. Chang, and S. Lyu. In ictu oculi: Exposing ai created fake videos by detecting eye blinking. In *WIFS*, pages 1–7. IEEE, 2018. 1, 2

[18] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 2

[19] J. Lorenzo-Trueba, J. Yamagishi, T. Toda, D. Saito, F. Villavicencio, T. Kinnunen, and Z. Ling. The voice conversion challenge 2018: Promoting development of parallel and non-parallel methods. In *Odyssey 2018 The Speaker and Language Recognition Workshop*, pages 195–202, 2018. 1, 2

[20] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010. 4

[21] H. H. Nguyen, T. Tieu, H.-Q. Nguyen-Son, V. Nozick, J. Yamagishi, and I. Echizen. Modular convolutional neural network for discriminating between computer-generated images and photographic images. In *ARES*, page 1. ACM, 2018. 2

[22] H. H. Nguyen, J. Yamagishi, and I. Echizen. Capsule-forensics: Using capsule networks to detect forged images and videos. In *ICASSP*. IEEE, 2019. 2, 3

[23] W. Quan, K. Wang, D.-M. Yan, and X. Zhang. Distinguishing between natural and computer-generated images using convolutional neural networks. *IEEE Transactions on Information Forensics and Security*, 2018. 2

[24] R. Raghavendra, K. B. Raja, S. Venkatesh, and C. Busch. Transferable deep-CNN features for detecting digital and print-scanned morphed face images. In *ICCV Workshop*. IEEE, 2017. 2

[25] N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen. Distinguishing computer graphics from natural images using convolution neural networks. In *WIFS*. IEEE, 2017. 1, 2

[26] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*, 2018. 1, 2, 4, 5

[27] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. Faceforensics++: Learning to detect manipulated facial images. *arXiv preprint arXiv:1901.08971*, 2019. 1, 4, 5, 6

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5

[29] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman. Synthesizing obama: learning lip sync from audio. *ACM Transactions on Graphics*, 2017. 1, 2

[30] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner. Face2Face: Real-time face capture and reenactment of RGB videos. In *CVPR*. IEEE, 2016. 1, 2, 4

[31] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis. Two-stream neural networks for tampered face detection. In *ICCV Workshop*. IEEE, 2017. 2

[32] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis. Learning rich features for image manipulation detection. In *CVPR*, pages 1053–1061, 2018. 2