Robust Manipulation Parameter Estimation Scheme for Post-JPEG Compressed Images using CNN

Vijayakumar Kadha*[1], Santos Kumar Das*[2]
*Department of ECE, NITR, Rourkela, Odisha, India
Email:{[1]vijayakumar_kadha, [2]dassk}@nitrkl.ac.in

***Abstract*—The significance of estimating the manipulation parameters in forged images has grown to grasp the life cycle of the image and gives forensic clues. Although several techniques have been put forth to do this, they are limited to a single alteration in the uncompressed scenario. However, estimation is difficult for multiple manipulations because a new image modification is produced every time. Therefore, it is necessary to repeat the procedure of creating a mathematical framework to determine a parameter estimation. To get around this, we create a deep residual framework that can estimate multiple manipulation parameters in a post-JPEG compressed image with achievable arbitrary precision. Specifically, the framework goes through the steps of noise residual extraction, extraction of features, and classification. To begin, the front-end detector is greatly augmented by the noise residual extraction stage, which adds three residual blocks with skip connections to generate noise residuals by reducing visual features and boosting alteration clues. Further, characteristics of the deep tampering clues are then obtained using a cross-feature learning technique and procured to fully connected layers for classification. To improve forged parameter estimates in the real world, we conduct experiments on three modifications and then apply several quality factors to the results. To top it all off, the proposed MPeR-Net outperforms state-of-the-art methodologies in the most crucial instance of lossy post-JPEG compression.**

***Index Terms*—Digital image forensics, manipulation parameter estimation, post-JPEG compression, forgery detection.**

I. Introduction

Nowadays, images, videos, and text are the most prevalent digital data in our day-to-day lives. However, the authenticity and integrity of digital data are significantly impacted by advanced altering programs that are freely available online. In some circumstances, such as journalistic integrity and investigative techniques, falsified photographs might harm society and our day-to-day activities. Therefore, to nail how each editing process was used to characterize entire life cycle of an image, the topic of digital image forensics was introduced [1]. Over the past decade, many forensic methods such as image splicing [2], [3], image resampling [4], [5], copy-move, and [6], [7], inpainting [12], [13] are contemplated to spot the trustworthiness of digital data. To create a forged image, the forger must select a rescaling factor, an Additive White Gaussian Noise (AWGN) variance parameter, a quality factor, and blur parameters (kernel size) to resample, noise addition, compress, and smooth an image, respectively. However, very few techniques focus on manipulation detection and its parameter estimation of faked images in the uncompressed scenario [4]–[11], [13], [16], [20]–[25]. Some traditional methods frequently build a theoretical model to describe a modified image by estimating manipulation parameters. Therefore, these parameters can be reversed to find the effects of editing, and information about an image original state can be given to an investigator. Hence, they can be utilized for camera model identification [18], stenographic algorithms, [32] and watermark detectors [33]. Therefore, estimating manipulation parameters is crucial for various additional forensic and security-related operations. On the other hand, JPEG is one of the most widely used image formats worldwide, and altered images are frequently stored in this format [25], [32]. When spotting the usage of various editing operations and tracing processing chains, it may be advantageous or even crucial to discover the manipulation parameter values [9], [23]. Therefore, it makes sense to consider the manipulation in a lossy compression scenario.

We proposed a unique method for Manipulation Parameter estimation in a Re-compressed images Network (MPeR-Net) to tackle this issue. Here, interpreting and anticipating the noise that JPEG lossy compression creates is the most important step in mitigating the detrimental impacts of compression. Additionally, MPeR-Net is an end-to-end solution that does not require pre/post-processing operations. The significant research contributions of this work are listed as follows.

- A unique end-to-end deep residual framework has been developed to get the best performance when estimating parameters for image alteration followed by recompression.
- MPeR-Net noise residuals are pulled from the modified images by concealing their texture details and enriching the manipulation traces.
- The proposed MPeR-Net robustness against JPEG post-compression is demonstrated by a comparison with reported deep learning techniques, where it is found to perform exceptionally well.

The remainder of the article is organized as follows. Section II explains the most contemporary research works for manipulation parameter estimation in compressed and un-compressed domains. In section III the suggested residual framework architecture and implementation details are described. Further, testing results and performance investigations are discussed in section IV. Conclusions and forthcoming directions are concluded in the last section.

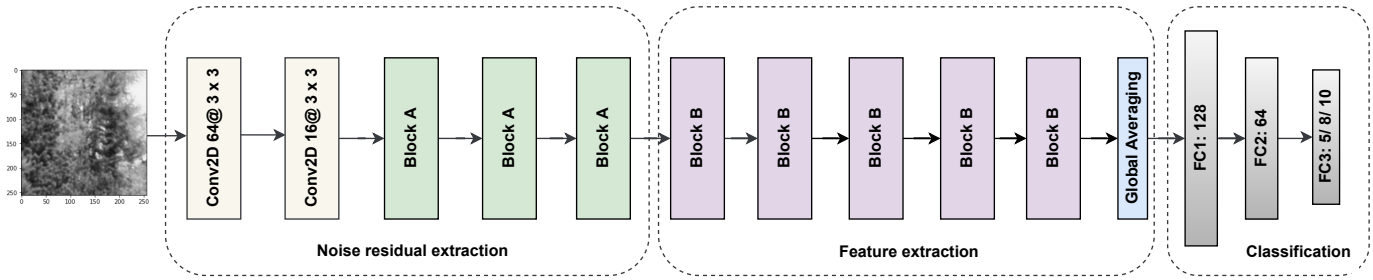


Fig. 1: The MPeR-Net framework for manipulation parameter estimation in re-compressed images.

II. Related work

We were motivated by several recent studies that explored inventive attempts to acquire rich patterns for image alteration detection and estimation (see Table I for further information). Forensic systems based on data-driven approaches have recently been presented by researchers as a method for end-to-end training that may not need specific characteristics or preprocessing processes. Hence, we concentrate on deep learning methods to estimate the parameters for the median filtering (MF), Gaussian Blurring (GB), AWGN addition (GN), and resampling (UP & DOWN) manipulations.

Initially, manipulation detection or estimation clues have been achieved from noise residuals which can be accomplished by steganalysis-rich model (SRM) kernels [14]. On the other hand, a high pass filter as preprocessing was used to recognize fake faces [20], and further, the authors [15] presented a CNN-based median filtering detection methodology using median filter residual from given data. Later, Bayar and Stamm proposed MISLNet, which drives the CNN to acquire prediction error filters. Above mentioned models can accurately categorize a wide range of modifications employed on uncompressed images; therefore, the research works [5], [24] is proposed for multiple manipulations. In addition, in [24], a manipulation tracing network containing abnormal deep features for fraud detection and localization by retrieving the manipulation traces using the constraint convolutional layers [5] and the SRM kernel [14] in uncompressed images. Additionally, Table I summarises the most important data-driven algorithms to estimate image manipulation parameters ('+': Estimation, '*': Detection '-': Neither Detection nor Estimation, PP?: Post-JPEG Compression). A detailed review is also available [17].

According to the literature, there are a few limitations that can be observed: (1) Most of the estimators are implemented for uncompressed images; however, their performance degrades significantly for the compressed scenario; and (2) the majority of forensic approaches focus on a single type of manipulation parameter estimation. Based on the shortcomings mentioned earlier, we proposed an MPeR-Net to categorize multiple manipulation types and parameters, which are provided in Table III, assuming that JPEG compression has been carried out using a variety of compression factors.

TABLE I: Overview of latest image manipulation parameter estimation schemes

Method	Clue/Feature	DNN Type	Manipulation parameters					PP?
			Blurring		AWGN	Resampling		
			MF	GB	GN	UP	DOWN	
[10]	Aritifacts	VGG Net	-	-	-	+	+	+
[21]	Arificats	Alex Net	+	+	+	+	+	-
[25]	Compression	Resnet	*	*	*	*	*	+
[22]	Denoiser	VGG Net	-	-	-	+	+	-
[16]	Aritifacts	VGG Net	+	-	-	-	-	-
MPeR-Net	Artifacts	ResNet	+	+	+	+	+	+

III. Proposed Method

This section depicts a precise outline of our proposed MPeR-Net, architecture details, and implementation.

A. Outline

The MPeR-Net framework looks at the estimation problem as classification, and it undergoes three phases, such as noise residual extraction, deep features extraction, and classification, as depicted in Fig.1. Pooling layers are like low-pass filters that localize the image's texture information. However, the forged image is scene-independent and needs to localize the generated region. Therefore, to extract noise residuals from inspected images, we utilized two convolutional layers followed by three residual blocks (see Fig. 2(a)) with skip connections without pooling layers. Which was motivated by SRNet [19] implemented in steganalysis. Next, to learn manipulation clues from noise residuals effectively, we utilized five residual blocks (see Fig. 2(b)) for a multi-scale feature learning strategy. This results in considerable betterment in the manipulation detection and parameter estimation sensitivity. Finally, deep convolutional attributes are fed to the classification phase to estimate manipulation parameters in the post-JPEG compressed scenario.

B. MPeR-Net Architecture

The proposed MPeR-Net, consists of two convolutional layers (l_{c_1}, l_{c_2}) followed by three residual blocks ($l_{r_1}, l_{r_2}, l_{r_3}$) with skip connections without pooling layers. Further, five residual blocks ($l_{r_4}, l_{r_5}...., l_{r_8}$) including max-pooling layers with skip connections and followed by global average pooling and three fully connected layers ($l_{fc_1}, l_{fc_2}, l_{fc_3}$). All convolutional layers in the proposed MPeR-Net utilized kernel with a size of 3×3. The detailed architecture, residual block's output feature shape, and total trainable parameters are shown

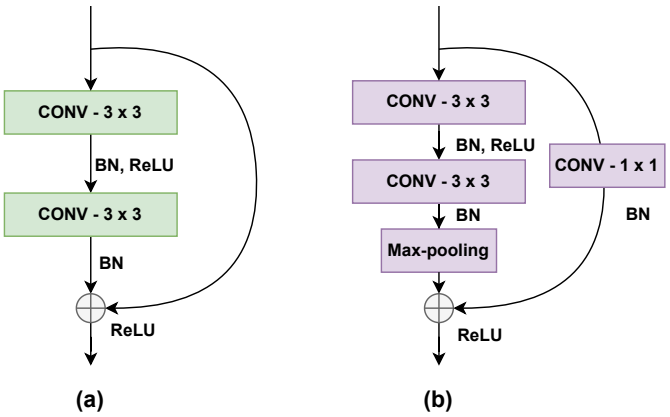


Fig. 2: Proposed MPeR-Net building blocks (a) Block A (b) Block B.

TABLE II: MPeR-Net architecture & total trainable parameters

Layer	Output Shape	Parameters
Input	(None, 256, 256, 1)	0
Conv2D (l_{c_1})	(None, 254, 254, 64)	640
Conv2D (l_{c_2})	(None, 252, 252, 16)	9232
BlockA (l_{r_1})	(None, 254, 254, 16)	4640
BlockA (l_{r_2})	(None, 252, 252, 16)	4640
BlockA (l_{r_3})	(None, 252, 252, 16)	4640
BlockB (l_{r_4})	(None, 126, 126, 16)	4912
BlockB (l_{r_5})	(None, 63, 63, 64)	47296
BlockB (l_{r_6})	(None, 32, 32, 128)	2297960
BlockB (l_{r_7})	(None, 16, 16, 256)	918272
BlockB (l_{r_8})	(None, 8, 8, 512)	3671552
Global.Avg.Pool	(None, 512)	0
Dense (l_{fc_1})	(None, 128)	65664
Dense (l_{fc_2})	(None, 64)	8256
Dense (l_{fc_3})	(None, 10)	510
Total Parameters	4965374	
Trainable Parameters	4965374	

in Table II. Finally, the output layer l_{fc_3} with 5, 8, 8, and 10 nodes represents Gaussian filter kernel size, median filter kernel size, AWGN variance, and scaling factor parameter estimations, respectively, in the post-JPEG compressed scenario. In the MPeR-Net, activation function (ReLU [27]) and Batch normalization [26] are introduced and followed after every convolutional layer.

C. Implementation details

The proposed MPeR-Net hyper-parameters such as filter weights for all convolutional ($l_{r_1}, l_{r_2}, ..l_{r_8}$) and fully connected layers ($l_{fc_1}, l_{fc_2}, l_{fc_3}$) are initialized with the He initializer [28]. For l_{fc_1} layer, the L2 kernel regularization with 0.01 is employed, whereas there is no regularisation for the other layers. All layers biases are set to 0.2. Initially, MPeR-Net was trained for 20 epochs with a learning rate (l_r) of 0.001, and then, it was trained by an additional 10 epochs with a lower l_r of 0.0001 to boost performance. Additionally, we employed a categorical cross-entropy loss function for all experiments.

TABLE III: Different manipulations, algorithms & its parameters to create datasets

Manipulation	Algorithm	Parameter	Parameter Values
Image resampling	Bilinear	Scaling factor	0.5, 0.6,....1.4, 1.5
	Bicubic	Scaling factor	0.5, 0.6,....1.4, 1.5
Noise addition	AWGN	Variance	0.1,0.3,0.6,0.9,1.2,1.5,1.8
Image blurring	Median filter	Kernel size	3, 5, 7, 9, 11, 13, 15
	Gaussian blur	Kernel size	3, 7, 11, 15

IV. Results & Discussions

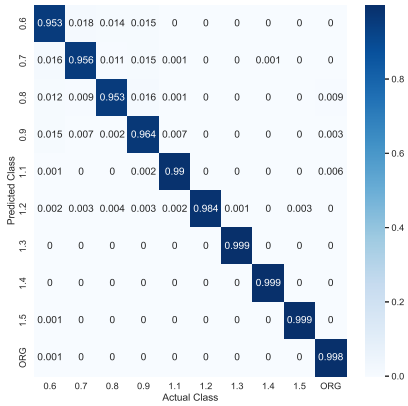
This section describes the dataset creation and performance analysis of the MPeR-Net is demonstrated by performing several experimentations by assuming different manipulations to estimate its parameters for post-JPEG compression images. In addition, the data processing is done using Python's programming language, and the MPeR-Net is developed using the Keras framework with GPU functioning as the background. The workstation used for implementation contains a 24 GB NVIDIA Quadro P6000 GPU, an 18 CUDA core Intel Xeon 6140 processor, and 256 GB of RAM.

A. Dataset Creation

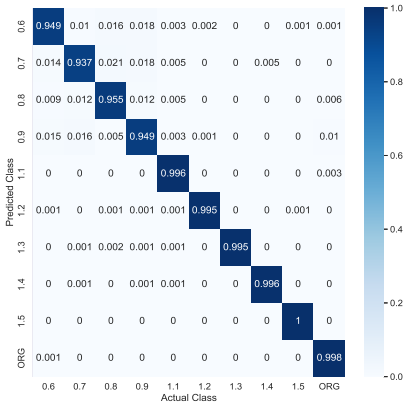
Implementing the proposed MPeR-Net was measured by conducting experiments on images taken from heterogeneous datasets such as RAISE [29] and DRESDEN [30]. 3,000 unprocessed images with 4928×3264 resolution are taken from the RAISE for creating a training dataset. Ten 512×512 non-overlapping patches are retained from each image to produce 30,000 image patches. Next, built-in camera compression settings are performed via JPEG compression by adjusting the primary quality factor value in the 95–97. Further, these images are altered (i.e., Image blurring, Noise addition, Image resampling) with different parameters to create pre-compressed manipulated images. In addition to this, these manipulated images are re-compressed to form altered post-JPEG compressed images.

Firstly, for an image blurring manipulation, each image of size 512×512 is manipulated with two different filterings (i.e., median filtering, Gaussian blurring) with various sizes of the kernel as described in Table III; then, it is post-JPEG compressed randomly picked secondary quality factor (i.e., QF_2 = 50 to 90) to form post-JPEG compressed images. Next, the middle segment of size 256×256 is trimmed to create forged images, and its interrelated unaltered post-JPEG compressed images are also taken to create original images. Further, two datasets are formed to estimate Gaussian and median filtering kernel size-dependent variance.

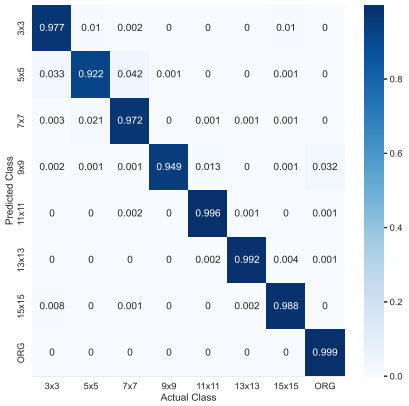
Similarly, for an AWGN noise manipulation, one database was created with six classes, five altered with different variances, and one unaltered image. Finally, for image resampling manipulation, two databases are created for bilinear and bicubic interpolation with ten scaling factors (upsampling & downscaling) followed by re-compression. Each database consists of ten altered classes and one unaltered class. These datasets are used to accomplish three different investigations to inspect the performance of the MPeR-Net.



(a) Scaling factor estimation (bicubic interpolation)



(b) Scaling factor estimation (bilinear interpolation)



(c) Median filter kernel size estimation

Fig. 3: Interpretation of the MPeR-Net in terms of confusion matrix with ten classes for scaling factor (bicubic & bilinear) estimation and eight classes for median filter kernel size estimation with post-compression quality factor ($QF_2 = 90$).

B. Resampling followed by Re-compression: Scaling parameter estimation

Image resampling enlarges or decreases its resolution by creating a new representation of the image having different pixel widths or heights. Therefore, we used the upscaling (UP), and downscaling (DOWN) factors ranging from 0.5 to 1.5 on the actual image to cover the growing and shrinking image sizes of its original resolution.

Performance of the proposed MPeR-Net measured by considering two most common interpolation techniques (i.e., bilinear and bicubic) to differentiate manipulated images with multiple scaling factors and unaltered images. For this experiment, the last fully connected layer consists of 10 nodes representing ten scaling factors to create altered images, and remaining nodes indicate unaltered images class, respectively. For training MPeR-Net, 300,000 image patches are considered with a size of 256×256 for 10 classes and 30,000 patches for each class. Further, MPeR-net was trained for 30 epochs with Adamax optimizer and categorical cross-entropy loss. In addition, to test the significance of the proposed MPeR-net, 600 images were collected from DRESDEN [30], unseen by the model, to generate 60,000 image patches to form testing sets with 6,000 patches of each class. Further, the proposed MPeR-net is tested on the above-specified dataset, and the corresponding confusion matrix of scaling factor estimation for bicubic and bilinear are depicted in Fig. 3 (a) & (b), respectively. From the observations, MPeR-net can effectively estimate upscaling factors of almost 99.99% in the post-JPEG compression scenario. In addition, a downscaling factor of 0.5 was calculated with higher accuracy of 95.25%, which was a challenging scenario in the post-JPEG compression scheme.

C. AWGN noise addition followed by Re-compression: Variance of noise estimation

The term "noise addition" describes the insertion of high-frequency information (also known as "noise-like signals"), which has no bearing on the content of images. To achieve this, Gaussian noise (GN) is independently injected into each pixel to hide the traces of manipulations. To perform this experiment, we distinguish between manipulated images created by adding AWGN with seven variances, as presented in Table III, and unaltered images in the post-JPEG compression scenario. In addition to this, last fully connected layer in the proposed MPeR-Net with eight nodes is taken to estimate the noise variance parameter. Further, 240,000 image patches are collected with a size of 256×256 from RAISE, [29] including 30,000 for each class. Next, the proposed MPeR-Net is trained with dataset mentioned earlier for 30 epochs by an Adamax optimizer with categorical cross-entropy as a loss function. For test MPeR-Net, 48,000 image patches are collected from DRESDEN, [30] with 6,000 for each class. Finally, trained model is tested on the aforementioned dataset to check cross-dataset settings, and corresponding confusion matrix of the MPeR-Net to estimate AWGN variance is shown in Fig. 4. By observing it, MPeR-Net can effectively estimate the AWGN noise parameter followed by re-compression with accuracy of above 99.10%.

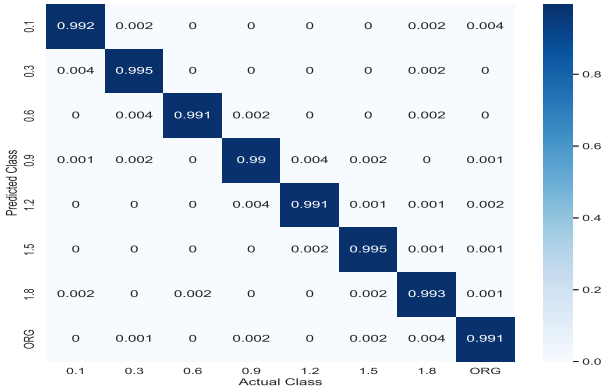
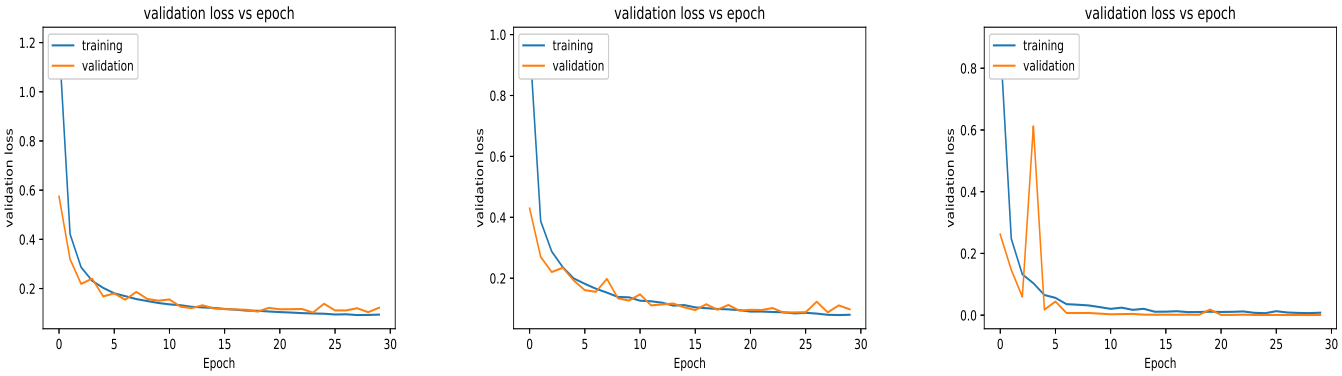


Fig. 4: Confusion matrix for estimation of noise variance manipulated using AWGN followed by Re-compression.



(a) Median filter kernel size estimation (b) Gaussian filter kernel size estimation (c) AWGN noise variance estimation

Fig. 5: Interpretation of the MPeR-Net in terms of loss with eight classes for median filter kernel size estimation and five classes for gaussian filter kernel size estimation with post-compression quality factor ($QF_2 = 90$).

D. Image blurring followed by Re-compression: Kernel size estimation

Blurring an image is a common technique for filtering out noise at high frequencies. Median and Gaussian filtering are two types a forger could use to hide their manipulation's fingerprints.

1) Median filter kernel size estimation: Reducing resampling artifacts can be achieved by employing a non-linear filter such as a median filter (MF) [31]. Primarily, it is used as an anti-forensics technique. However, median filter parameter estimation is difficult in a re-compression scenario because compression artifacts can erase traces left by median filtering. To tackle this issue, we experimented with eight classes, where seven are manipulated with median filtering and one unaltered type. The training dataset is created as in the previous experiment, with different median filter kernel sizes to create altered images. A total of 240,000 patches, with

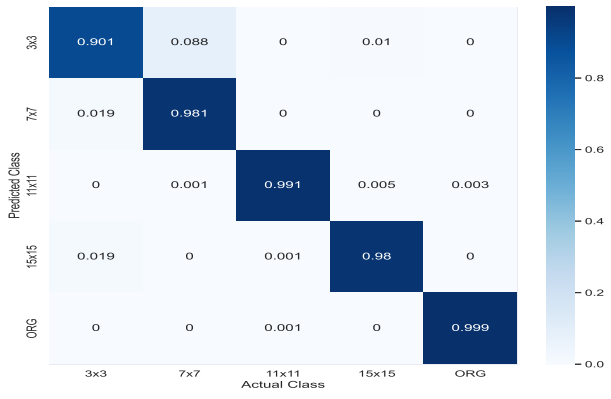


Fig. 6: Confusion matrix of kernel size estimation for manipulated using Gaussian filter followed by Re-compression.

30,000 for each class, are created, and the proposed MPeR-Net is trained on RAISE [29] and tested on DRESDEN [30], and observed that median filtered kernel size is estimated at more than 92.20%. The confusion matrix of the proposed MPeR-Net for kernel size estimation of median filtering followed by re-compression is shown in Fig. 3(c).

TABLE IV: Comparative analysis of manipulation estimation parameters in post-JPEG compressed scenario.

Method	Manipulation parameter estimation				
	Blurring		AWGN	Resampling	
	MF	GB	GN	UP	DOWN
MISLnet [21]	96.84%	**97.68%**	-	98.15%	91.24%
CNN-RFE [22]	-	-	-	95.84%	89.46%
Dual-Net [10]	-	-	-	96.36%	90.65%
MPeR-Net	**97.15%**	96.94%	**99.12%**	**99.56%**	**94.75%**

2) Gaussian filter kernel size estimation: The proposed MPeR-Net effectively pulls the manipulation traces from the previous investigation to estimate the median filter kernel size in the post-JPEG compressed scenario. In addition, a linear filter such as Gaussian blur (GB) is used to suppress noise by reducing high-frequency signals. Hence, we considered estimating the gaussian filter kernel size, which depends on variance. For this experiment, 150,000 image patches with 30,000 for each class are created from the RAISE [29] to train the proposed MPeR-Net and tested on 30,000 images created from DRESDEN [30]. From the observations, the Gaussian filter kernel size is estimated at more than 90.10%. The confusion matrix of the proposed MPeR-Net to estimate gaussian filter kernel size estimation for re-compressed images is shown in Fig. 6.

E. Baseline methods for comparison

For comparative analysis, we took the previous experiment environments for recent-baseline methods and encountered the average accuracy of manipulation parameter estimation of post-JPEG compressed images. For the comparative study, we utilised the earlier experiment environments for more modern baseline approaches. Throughout this process, we encountered the typical accuracy of manipulation parameter estimates for post-JPEG compressed image data. The performance of the MPeR-Net is analyzed and compared with existing baseline

methods such as MISLnet [21], CNN-RFE [22], and Dual-Net [10]. Further, it has a higher mean accuracy of median filtering kernel size, AWGN addition, upscaling, and downscaling factors with 97.15%, 99.12%, 99.56%, and 94.75%, respectively, than the previously reported techniques. In addition, MPeR-Net performs better downscaling followed by re-compression, as shown in Table IV. However, proposed MPeR-Net often has trouble differentiating between Gaussian filtering with a kernel size of 3×3 and 7×7.

V. Conclusions

This paper introduced MPeR-Net, a novel deep residual framework for estimating manipulation parameters in post-JPEG compressed images. Firstly, MPeR-Net gathers image-modifying fingerprint attributes from a query image, and further, it examines how distinct the attained characteristics are from its benchmark fingerprints to find alteration cues. The noise residual extraction phase of the proposed framework effectively pulls the forgery clues by extending the front-end detector, which can take advantage of noise residuals by minimizing the texture information of the image. Furthermore, using shortcut connections from noisy residuals, MPeR-Net efficiently predicts manipulation parameters. In addition, our network outperforms recently reported techniques, specifically in the JPEG domain. In subsequent work, we will create a comprehensive framework to defence against malicious attacks and investigate the correctness of the proposed MPeR-Net for further post-processing methods, such as contrast enhancement and image morphing.

References

[1] Stamm, Matthew C., Min Wu, and KJ Ray Liu. "Information forensics: An overview of the first decade," IEEE access 1 (2013): 167-200.

[2] S. Lyu, X. Pan, and X. Zhang,"Exposing region splicing forgeries with blind local noise estimation," International Journal of Visual Communication and Image Representation, vol. 110, no. 2, pp. 202–221, Nov. 2014.

[3] M. Huh, A. Liu, A. Owens, and A. A. Efros, "Fighting fake news: Image splice detection via learned self-consistency," in Proceedings European Conference on Computer Vision, 2018, pp. 101–117.

[4] D. Vazquez-Padin, F. Perez-Gonzalez, P. Comesana-Alfaro, "A random matrix approach to the forensic analysis of upscaled images," IEEE Transactions on Information Forensics and Security 12 (9) (2017) 2115–2130.

[5] B. Bayar, M. C. Stamm, "Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection," IEEE Transactions on Information Forensics and Security 13 (11) (2018) 2691–2706.

[6] Y. Li and J. Zhou, "Fast and effective image copy-move forgery detection via hierarchical feature point matching," IEEE Transactions on Information Forensics and Security, vol. 14, no. 5, pp. 1307–1322, May 2019.

[7] J.-L. Zhong and C.-M. Pun, "An end-to-end dense-InceptionNet for image copy-move forgery detection," IEEE Transactions on Information Forensics and Security, vol. 15, pp. 2134–2146, 2020.

[8] X. Liu, W. Lu, Q. Zhang, J. Huang and Y. -Q. Shi, "Downscaling Factor Estimation on Pre-JPEG Compressed Images," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 3, pp. 618-631, March 2020.

[9] Liu, X., Lu, W., Xue, Y. et al. "Upscaling factor estimation on double JPEG compressed images." Multimed Tools Appl 79, 12891–12914 (2020).

[10] L. Peng, X. Liao, M. Chen,"Resampling parameter estimation via dual-filtering based convolutional neural network," Multimedia Syst. (2020).

[11] Luo, S., Luo, J., Lu, W., Fang, Y., Zeng, J., Shi, S.S., Zhang, Y., 2021. "Resampling factor estimation via dual-stream convolutional neural network," Comput. Mater. Continua (CMC) 66 (1), 647–657

[12] H. Li, W. Luo, and J. Huang, "Localization of diffusion-based inpainting in digital images," IEEE Transactions on Information Forensics and Security, vol. 12, no. 12, pp. 3050–3064, Dec. 2017.

[13] A. Li et al., "Noise doesn't lie: Towards universal detection of deep inpainting," in Proc. 13th Int. Joint Conf. Artif. Intell., Aug. 2021, pp. 1–7.

[14] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," IEEE Transactions on Information Forensics and Security, vol. 7, no. 3, pp. 868-882, Jun. 2012.

[15] J. Chen, X. Kang, Y. Liu, Z. J. Wang, "Median filtering forensics based on convolutional neural networks," IEEE Signal Processing Letters 22 (11) (2015) 1849–1853.

[16] J. Zhang, Y. Liao, X. Zhu, H.Wang, and J. Ding, "A deep learning approach in the discrete cosine transform domain to median filtering forensics," IEEE Signal Process. Lett., vol. 27, pp. 276-280, 2020.

[17] L. Verdoliva, "Media Forensics and DeepFakes: An Overview," in IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 5, pp. 910-932, Aug. 2020.

[18] O. Mayer and M. C. Stamm. "Learned forensic source similarity for unknown camera models," In IEEE International Conference on Acoustics, Speech and Signal Processing, 2018.

[19] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images," IEEE Transactions on Information Forensics and Security, vol. 14, no. 5, pp. 1181–1193, May 2019.

[20] H. Mo, B. Chen, and W. Luo, "Fake faces identification via convolutional neural network," in Proc. 6th ACM Workshop Inf. Hiding Multimedia Secur., Jun. 2018, pp. 43-47.

[21] Belhassen Bayar and Matthew C. Stamm. 2017. "A Generic Approach Towards Image Manipulation Parameter Estimation Using Convolutional Neural Networks," In Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security (IH & MMSec '17). Association for Computing Machinery, New York, NY, USA, 147–157.

[22] Chang Liu and Matthias Kirchner. 2019. "CNN-based Rescaling Factor Estimation," In Proceedings of the ACM Workshop on Information Hiding and Multimedia Security (IH & MMSec'19). Association for Computing Machinery, New York, NY, USA, 119–124.

[23] Zhang, Qin, Wei Lu, Tao Huang, Shangjun Luo, Zhaopeng Xu, and Yijun Mao. "On the robustness of JPEG post-compression to resampling factor estimation," Signal Processing 168 (2020): 107371.

[24] Y. Wu, W. Abdalmageed, and P. Natarajan, "ManTra-Net: Manipulation tracing network for detection and localization of image forgeries with anomalous features," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 9543-9552.

[25] I.-J. Yu, S.-H. Nam, W. Ahn, M.-J. Kwon, H.-K. Lee, "Manipulation classification for jpeg images using multi-domain features," IEEE Access 8 (2020) 210837–210854.

[26] S. Ioffe, C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in: International conference on machine learning, PMLR, 2015, pp. 448–456.

[27] Agarap, Abien Fred. "eep Learning using Rectified Linear Units (ReLU)," ArXiv abs/1803.08375 (2018)

[28] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026-1034.

[29] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, G. Boato, "Raise: A raw images dataset for digital image forensics," in: ACM Multimedia Systems Conference, MMSys '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 219224.

[30] T. Gloe, R. Bhme, "The 'Dresden Image Database' for benchmarking digital image forensics," in: Symposium On Applied Computing, Vol. 2, 2010, pp. 1585–1591.

[31] Böhme, Rainer, and Matthias Kirchner. "Counter-forensics: Attacking image forensics," In Digital image forensics, pp. 327-366. Springer, New York, NY, 2013.

[32] T. Pevny and J. Fridrich, "Detection of double-compression in JPEG images for applications in steganography," IEEE Transactions on Information Forensics and Security, vol. 3, no. 2, pp. 247–258, June 2008.

[33] Ruanaidh, J. J. O., and Pun, T. "Rotation, scale and translation invariant spread spectrum digital image watermarking," Signal processing 66, 3 (1998), 303–317.