ON THE ROBUSTNESS OF CONSTRAINED CONVOLUTIONAL NEURAL NETWORKS TO JPEG POST-COMPRESSION FOR IMAGE RESAMPLING DETECTION

Belhassen Bayar and Matthew C. Stamm

Department of Electrical and Computer Engineering,
Drexel University, Philadelphia, PA 19104

ABSTRACT

Detecting image resampling in re-compressed images is a very challenging problem. Existing approaches to image resampling detection operate by building pre-selected model to locate periodicities in linear predictor residues. Additionally, if an image was JPEG compressed before resampling, existing techniques detect tampering using the artifacts left by the pre-compression. However, state-of-the-art approaches cannot detect resampling in re-compressed images initially compressed with high quality factor. In this paper, we propose a novel deep learning approach to adaptively learn resampling detection features directly from data. To accomplish this, we use our recently proposed constrained convolutional layer. Through a set of experiments we evaluate the effectiveness of our proposed constrained convolutional neural network (CNN) to detect resampling in re-compressed images. The results of these experiments show that our constrained CNN can accurately detect resampling in re-compressed images in scenarios that previous approaches are unable to detect.

Index Terms— Image forensics, convolutional neural networks, constrained convolutional layer, deep convolutional features.

1. INTRODUCTION

Identifying the processing history of an image is an important task in image forensics. Many image manipulations leave behind unique traces that can be used to detect the type of image editing. Thus, researchers proceeded by extracting image manipulation features from these traces then develop associated algorithms to determine the type of processing operation [1]. Numerous forensic techniques have been developed to detect several manipulations such as median filtering [2, 3], contrast enhancement [4], blurring [5], etc.

Detection of image resampling has drawn particular attention. Most techniques to perform image resampling detection operate by identifying traces in the residue of local linear predictor and exploit the periodic interpolation artifacts [6, 7]. More explicitly, Popescu and Farid [6] proposed to detect resampling by jointly estimating the prediction weights and a pixel dependency measure so called *p-map*. This approach is very computationally costly. To address this problem, Kirchner [7] proposed a simplified approach that operates by locating periodic variance of the prediction error.

While resampling detection is very effective in uncompressed images, many resampling traces are masked or destroyed when JPEG compression is applied to images after resampling has been performed. This is known as post-compression. Kirchner and Gloe [8] demonstrated that if an image was previously JPEG compressed before resampling, then shifted JPEG blocking artifacts that appear in an image's p-map can be used to detect resampling in re-compressed images (i.e., images that have undergone JPEG compression, followed by resampling, then JPEG compression again).

While this approach can successfully detect resampling if the first compression was performed using a low to moderate quality factor, it is still unable to detect resampling if the first compression was performed with a high quality factor [8]. This is particularly important since in most real world scenarios, images taken by digital cameras are saved using the camera's default compression settings that use very high quality factors. As a result, detecting resampling in re-compressed images with a high initial quality factor remains an important open problem.

Recently, a novel convolutional neural network (CNN) based approach using a new type of convolutional layer, called *constrained convolutional layer*, has been mainly designed to perform image forgery detection task [9]. In their standard form, CNNs tend to learn image's content which will lead to a classifier that detects objects and scenes in images. By contrast, the constrained convolutional layer has been used to suppress an image's content and adaptively learn image manipulation features. Instead of using pre-selected models or "hand-designed" features, this approach adaptively extracts features related to local pixel dependencies by forcing CNN to learn prediction-error filters while training the network.

In this paper, we propose a new CNN architecture that is able to perform image resampling detection in re-compressed images. To accomplish this, we use the constrained convolutional layer proposed in our recent work [9]. In our CNN architecture we use 1×1 convolutional filters to learn new associations between feature maps. Additionally, to improve the final accuracy of our proposed approach we use the deep convolutional features [10] learned by our CNN to train an extremely randomized trees classifier [11]. We evaluated the robustness of our CNN in extracting image resampling features from re-compressed images with different scaling and quality factors through a set of experiments. Our experimental results showed the effectiveness of our approach in detecting upscaling and downscaling tampering with re-compressed images.

2. CONVOLUTIONAL NEURAL NETWORKS

CNNs are an approach from deep learning that has gained attention due to their ability to learn classification features directly from data. They have been first proposed in the late 1980's with handwritten zip code recognition [12] as an extended version of neural networks (NN). Moreover, CNNs have been successfully used with a large variety of different types of signals such as speech [13], images [14]

This material is based upon work supported by the National Science Foundation under Grant No. 1553610. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

and text data [15].

In a CNN architecture, the convolutional layers act as features extractors. In fact, the input image to the network is first convolved with a set of parallel filters with a fixed dimension and an overlapping distance called stride. The output of each convolutional filter, known as feature map, is a new representation of the input data. Subsequently, these output feature maps are then convolved with the following hidden convolutional layers in the network to learn new lower level representation of the data. The output of the final convolutional layer is fed to a regular fully-connected neural network to perform the classification task. Similarly to NN, a convolutional layer is followed by an activation function to introduce non-linearity throughout the network.

The set of parallel convolutional operations yields a large feature map volume. Therefore, a convolutional layer is typically followed by a pooling layer for dimensionality reduction purpose. There exist many types of pooling layers such as max, average and stochastic pooling. These types of layers, retain the most significant features within a feature map. The max-pooling layer for instance, operates by keeping the maximum value within a sliding window with a stride distance.

The analytical expression of the convolutional operation between the input feature maps and a convolutional layer within the CNN architecture is given in Eq. (1):

$$\boldsymbol{h}_j^{(n)} = \sum_{k=1}^{K} \boldsymbol{h}_k^{(n-1)} * \boldsymbol{w}_{kj}^{(n)} + bj^{(n)}, \tag{1}$$

where $*$ denotes a $2D$ convolution, $\boldsymbol{h}_j^{(n)}$ is the j^{th} feature map output in the n^{th} hidden layer, $\boldsymbol{h}_k^{(n-1)}$ is the k^{th} channel in the $(n-1)^{th}$ hidden layer, $\boldsymbol{w}_{kj}^{(n)}$ is the k^{th} channel in the j^{th} filter in the n^{th} layer and $b_j^{(n)}$ is its corresponding bias term. The weights of the convolutional filters and fully connected layers are initially randomly seeded then updated during training through an iterative algorithm called back-propagation. This algorithm alternates between feedforward and back-propagation passes where the weights are updated during the back-propagation passes. At the end, we would like to minimize the average loss between the actual labels and the network outputs, i.e., $E = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{c} y_i^{*(k)} \log\left(y_i^{(k)}\right)$, where $y_i^{*(k)}$ and $y_i^{(k)}$ are respectively the true label and the network output of the i^{th} image at the k^{th} class with m training images and c neurons in the output layer.

To minimize the average loss, we use the stochastic gradient descent (SGD) solver [16]. The iterative update rules for the filters coefficients $\boldsymbol{w}_{ij}^{(n)}$ in CNN during the back-propagation pass is given below:

$$\begin{aligned} \nabla \boldsymbol{w}_{ij}^{(n+1)} &= m \cdot \nabla \boldsymbol{w}_{ij}^{(n)} - d \cdot \epsilon \cdot \boldsymbol{w}_{ij}^{(n)} - \epsilon \cdot \frac{\partial E}{\partial \boldsymbol{w}_{ij}^{(n)}} \\ \boldsymbol{w}_{ij}^{(n+1)} &= \boldsymbol{w}_{ij}^{(n)} + \nabla \boldsymbol{w}_{ij}^{(n+1)}, \end{aligned} \tag{2}$$

where $\boldsymbol{w}_{ij}^{(n)}$ is the i^{th} channel from the j^{th} kernel matrix in the n^{th} hidden layer that convolves with the i^{th} channel in the previous feature maps of the $(n-1)^{th}$ layer, $\nabla \boldsymbol{w}_{ij}^{(n)}$ denotes the gradient of $\boldsymbol{w}_{ij}^{(n)}$ and ϵ is the learning rate. The bias term $b_j^{(n)}$ in (1) is updated using the same equations presented in (2). For fast convergence as explained by LeCun *et al.* in [17], we use the *decay* and *momentum* strategy which are respectively denoted by d and m in (2).

3. CONSTRAINED CONVOLUTIONAL LAYER

Though existing approaches to resampling detection have proven effective, they are still not capable to successfully identify manipulation traces in re-compressed images that were initially compressed with high quality factors. Particularly, state-of-the-art technique [7] uses pre-determined model of prediction error to identify resampling artifacts which may lead to decision features that may not capture all the traces left by resampling. By contrast, CNNs have proven powerful at extracting classification features from data. However, with digital images CNNs learn features related to the image content. Therefore, the constrained convolutional layer has been designed for multimedia forensics task to suppress an image's content and extract prediction residual features [9] by enforcing prediction error-filters constraints on the first convolutional layer while training the network. More explicitly, each of the K filters $\boldsymbol{w}_k^{(1)}$ in the first layer of the CNN have the following constraints placed on them:

$$\begin{cases} \boldsymbol{w}_k^{(1)}(0,0) = -1 \\ \sum_{\ell,m\neq 0} \boldsymbol{w}_k^{(1)}(\ell,m) = 1 \end{cases} \tag{3}$$

The training of the constrained CNN is summarized in the following algorithm:

Algorithm 1 Training algorithm for constrained convolutional layer

```
 1: Initilize w_k's using randomly drawn weights
 2: i=1
 3: while $i \leq max\_iter$ do
 4:     Do feedforward pass
 5:     Update filter weights through stochastic gradient
         descent and backpropagate errors
 6:     Set $\mathbf{w}_k(0,0)^{(1)} = 0$ for all K filters
 7:     Normalize $\boldsymbol{w}_k^{(1)}$'s such that $\sum_{\ell,m\neq 0} \boldsymbol{w}_k^{(1)}(\ell,m) = 1$
 8:     Set $\mathbf{w}_k(0,0)^{(1)} = -1$ for all K filters
 9:     i = i+1
10:     if training accuracy converges then
11:         exit
12: end
```

As a result of the above algorithm, the first convolutional layer adaptively extracts prediction-error features from input images then the next convolutional hidden layers learn new lower representation of these features. Thus, in this work we use the constrained convolutional layer to capture pixel value dependencies and adaptively learn features related to the periodicities in the prediction error generated by the resampling operation. As mentioned above, existing methods are not very successful in detecting resampling traces in pre-compressed images with high quality factors.

In fact, it is known that images are generally saved in JPEG format in digital cameras with high quality factors. Additionally, JPEG generates strong peaks in the *p-map*'s spectrum that are used to detect resampling by identifying shifts in their positions [8]. The stronger the artifacts were, the more they will remain in the processed images. However, when an image is pre-compressed with high quality factor, the JPEG peaks are not anymore detectable. Moreover, a post-compression after resampling makes the detection even more difficult since the artifacts left by the first compression may be destroyed by the second one. On the other hand, our proposed constrained CNN adaptively extracts resampling features from data that may not be captured using pre-selected models. Thus, this approach generalizes a substantial amount of research in resampling detection.

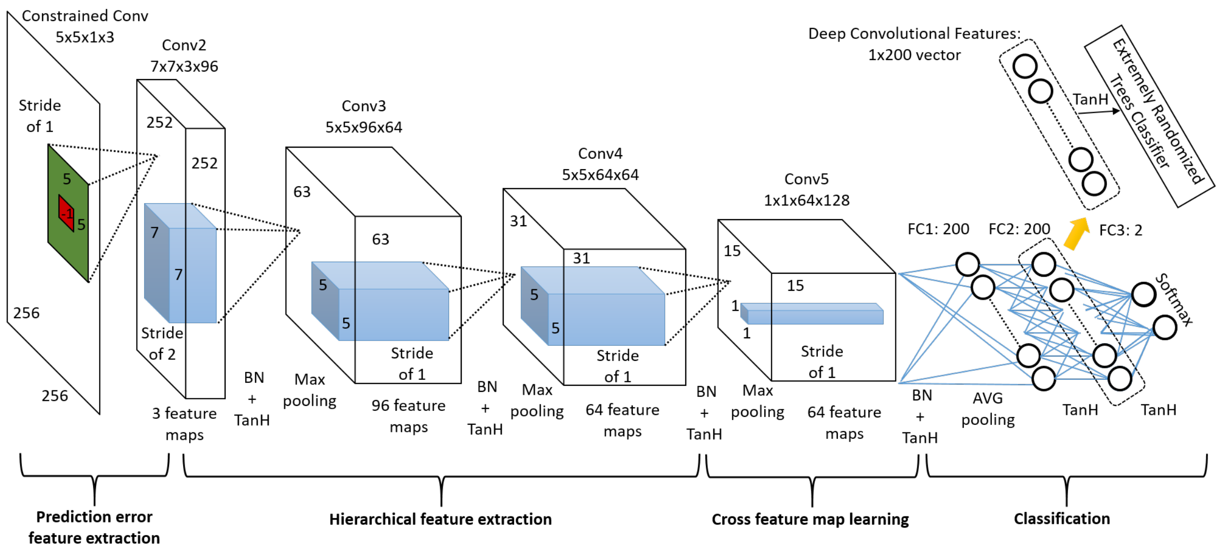

Fig. 1. CNN proposed architecture; BN:Batch-Normalization Layer; TanH: Hyperbolic Tangent Layer

4. NETWORK ARCHITECTURE

In this section we present a brief description about every type of layer that we have used in our proposed CNN architecture. Fig. 1 depicts the overall design of our CNN with details about the size of each layer. One can notice that we use four different blocks to perform different tasks: First, the constrained convolutional filters are used to perform prediction error feature extraction. The second block, uses three convolutional layers to extract lower level hierarchical features. Subsequently, we use a block that consists of 1×1 convolutional filters to learn new associations between different feature maps. Finally, the output of the latter block is fed to the classification block which consists of three fully-connected layers. In this work, the input layer of our CNN is the green layer of an image patch sized 256×256 pixels.

Convolutional Layers The convolutional layers in CNNs act as feature extractors. From Fig. 1, we can notice that we use three different types of convolutional layers, namely one "Constrained Conv" layer which is the constrained convolutional layer presented in Section 3, three regular convolutional layers and 128 1×1 convolutional filters in "Conv5" to learn new association between feature maps. One can notice also that we used a stride of size 1 in all the convolutional layers except in "Conv2" where we used a stride of 2. Fig. 1 depicts the size of filters in each convolutional layer as well as the dimension of their corresponding output feature maps.

Activation Function Similarly to neural networks, convolutional layers are followed by an activation function. In our architecture we use hyperbolic tangent (TanH) activation functions. In Fig. 1, we can notice that all the regular convolutional layers are followed by a TanH activation function. However, feature maps learned by the "Constrained Conv" layer are not followed by a TanH layer. This is mainly because the learned prediction residual error features can easily be destroyed by many types of nonlinear operations. Finally, to classify our images, we use two neurons in the output layer, i.e., original versus resampled, with a softmax activation function where every input image corresponds to the highly activated neuron.

Batch Normalization To minimize the internal covariate shift, which is the change in the input distribution to a learning system, we have to apply a zero-mean and unit-variance transformation of the data while training the CNN model. To do this, we use a batch normalization layer after each regular convolution. More specifically, The input to each layer gets affected by the parameters of all previous layers and even small changes get amplified. Therefore, the batch normalization layer addresses this problem and improves the final accuracy of CNN.

Pooling From Fig. 1, we can notice that we use two types of pooling in our CNN, i.e., three max-pooling and one average-pooling after "Conv5" layer. In all pooling layers, we use a sliding window of size 3×3 and stride of 2. This type of layer is mainly used to reduce the dimension of the large feature map volumes hence accelerates the training process by reducing the computational cost. Moreover, the max-pooling layer retains the maximum value within the local neighborhood of the sliding window, whereas, the average-pooling layer retains the average in a local neighborhood.

Deep Convolutional Features Similarly to [10], we extract the output of the activation function from the second fully-connected layer "FC2" by doing a feedforward pass of the training and testing data after completing the training of our CNN. Therefore, each 256×256 patch in the training and testing data has its corresponding 200 features vector. Then, we train an extremely randomized trees classifier using the new collected data. This latter approach has proven to improve the final classification accuracy.

5. EXPERIMENTS

We evaluated the robustness and ability of our CNN in adaptively extracting resampling detection features through a set of experiments. We use our proposed CNN as a binary classifier to detect resampling operations in re-compressed images with different scaling fac-

Table 1. CNN resampling detection rate using Softmax layer and ET classifier on re-compressed images with 150% upscaling, 120% upscaling and 50% downscaling manipulations

	Post-compression Quality Factor					
Resampling Operation	**QF = 50**	**QF = 60**	**QF = 70**	**QF = 80**	**QF = 90**	**No Re-compression**
150% upscaling (Softmax)	91.22%	96.44%	97.88%	98.19%	99.52%	99.83%
150% upscaling (ET)	91.35%	96.65%	97.82%	98.26%	99.61%	99.99%
120% upscaling (Softmax)	84.08%	92.99%	95.73%	95.62%	99.30%	99.35%
120% upscaling (ET)	84.02%	93.24%	95.86%	95.98%	99.42%	99.54%
50% downscaling (Softmax)	83.74%	87.99%	89.21%	86.93%	92.35%	98.19%
50% downscaling (ET)	83.98%	88.31%	89.69%	87.22%	92.48%	98.21%

tors and JPEG compression quality factors. Additionally, we use the deep convolutional features learned by our CNN to train an extremely randomized trees (ET) classifier.

Data collection We collected $6,500$ images of size at least $2,688 \times 1,520$ from 12 different camera models. To make sure that our approach did not learn features related to camera device parameters, the training and testing images were collected from two separate sets of devices and saved using the camera's default compression settings. Though most cameras use proprietary quantization tables, the tables used by the cameras in these experiments are approximately equivalent to compression using a quality factor in the 95-97 range. In this paper we consider the image resampling processing using bilinear interpolation with three different scaling factors, i.e., 50%, 120% and 150%.

We converted each image to grayscale by retaining only its green color layer of each image. We then divided each grayscale image into 256×256 pixel blocks to create a set of 30,000 unaltered blocks. Next, we created several sets of resampled and recompressed images. To accomplish this, we first rescaled every unaltered block using each of three different scaling factors: 50%, 120%, and 150%. After this, we compressed each of the resampled images and their corresponding unaltered version with different quality factors.

In total, we created 18 different databases, i.e., three scaling factors $(50\%, 120\%, 150\%)$ with five different re-compression quality factors $(QF = 50, 60, 70, 80, 90)$ along with no re-compression operation for each type of resampling. Each database consisted of $60,000$ patches where $50,000$ patches (approximately 84%) were used for training and $10,000$ patches (approximately 16%) for testing.

Training parameters We set the training parameters of the stochastic gradient descent as follows: $momentum = 0.95$, $decay = 0.0005$, and a learning rate $\epsilon = 10^{-3}$ that decreases every $5,000$ iterations by a factor $\gamma = 0.5$. We set the batch size equal to 64 and the described binary models in this section are trained for $50,000$ iterations. The testing accuracy was recorded every $1,000$ training iterations.

Experimental results We used our proposed constrained CNN architecture to perform resampling detection on each set of resampled and re-compressed images. We conducted a set of two experiments. To accomplish this, we first trained our CNN with 18 different datasets to assess the ability of our proposed model in classifying resampled images with different scaling and compression parameters. Then we evaluated the deep convolutional features approach [10] using an ET classifier [11]. Table 1 shows the detection accuracies with different quality factors and scaling parameters respectively equal to 150%, 120% and 50% using a softmax layer and ET classifier. One can notice from Table 1 that with upscaling operations our CNN can typically achieve an accuracy higher than 91.22% with all the quality factors except for 120% upscaled images with $QF = 50$ which are detected with an accuracy equal to 84.08%. Though the detection accuracy decreases when using low post-compression's quality factor, our approach can still detect resampling even with a low post-compression quality factor such as 50.

As mentioned above, Kirchner and Gloe technique [8] fails to detect downscaling operation when the first compression's quality factor is very high. However, from Table 1 we can see that our approach successfully detect 50% downscaled images with an accuracy higher than 83.74% with all quality factors. We believe that our constrained CNN is able to learn subtle artifacts left by shifts in the original JPEG blocking grid. We can also notice that CNN is able to detect resampling in re-compressed images with $QF = 70$ better than detecting re-compressed images with $QF = 80$ for 120% upscaling and 50% downscaling.

We finally use the deep convolutional features to train an ET classifier as explained in Section 4. We can notice that this method has improved the final accuracy of our approach with all our databases except with 120% upscaled re-compressed images with $QF = 70$ and 50% downscaled re-compressed images with $QF = 50$ where the testing accuracies decreased respectively from 97.88% to 97.82% and from 84.08% to 84.02%.

6. CONCLUSION

In this paper, we have proposed a novel deep learning based approach to perform image resampling detection in re-compressed images. To accomplish this, we used a constrained convolutional layer to adaptively learn features for detecting traces left by resampling and JPEG pre-compression. We evaluated the performance of this approach using images resampled using three different scaling factors and re-compressed using five different JPEG quality factors. The results of these experiments show that our constrained CNN can accurately detect resampling in re-compressed images in scenarios that previous approaches are unable to detect.

7. REFERENCES

[1] M. C. Stamm, M. Wu, and K. J. R. Liu, "Information forensics: An overview of the first decade," *IEEE Access*, vol. 1, pp. 167–

200, 2013.

[2] M. Kirchner and J. Fridrich, “On detection of median filtering in digital images,” in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2010, pp. 754 110–754 110.

[3] X. Kang, M. C. Stamm, A. Peng, and K. J. R. Liu, “Robust median filtering forensics using an autoregressive model,” *IEEE Transactions on Information Forensics and Security,*, vol. 8, no. 9, pp. 1456–1468, Sep. 2013.

[4] M. C. Stamm and K. J. R. Liu, “Forensic detection of image manipulation using statistical intrinsic fingerprints,” *IEEE Trans. on Information Forensics and Security*, vol. 5, no. 3, pp. 492 –506, 2010.

[5] K. Bahrami and A. C. Kot, “Image tampering detection by exposing blur type inconsistency,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 2654–2658.

[6] A. C. Popescu and H. Farid, “Exposing digital forgeries by detecting traces of resampling,” *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 758–767, Feb. 2005.

[7] M. Kirchner, “Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue,” in *Proceedings of the 10th ACM Workshop on Multimedia and Security*, ser. MM&Sec ’08. New York, NY, USA: ACM, 2008, pp. 11–20.

[8] M. Kirchner and T. Gloe, “On resampling detection in recompressed images,” in *2009 First IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2009, pp. 21–25.

[9] B. Bayar and M. C. Stamm, “A deep learning approach to universal image manipulation detection using a new convolutional layer,” in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2016, pp. 5–10.

[10] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition.” in *ICML*, 2014, pp. 647–655.

[11] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely randomized trees,” *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[13] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition,” *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[15] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[16] H. Robbins and S. Monro, “A stochastic approximation method,” *The annals of mathematical statistics*, pp. 400–407, 1951.

[17] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, “Efficient backprop,” in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.