参赛队员姓名：陈思达

中学：上海市世界外国语中学

省份：上海市

国家/地区：中国

指导教师姓名：董临风

指导教师单位：中国福利会少年宫

论文题目：LBPNet: Inserting Local Binary Patterns into Neural Networks to Enhance Manipulation Invariance of Fake Face Detection

LBPNet: Inserting Local Binary Patterns into Neural Networks to Enhance Manipulation Invariance of Fake Face Detection

Chen Sida
Shanghai World Foreign Language Academy
sidachen2003@gmail.com

Abstract

Fake face detection is an essential yet under-explored area, because faces generated by modern Generative Adversarial Networks (GANs) are virtually indiscernible by human observers. A recent work inventing the GramNet [19] proposed using global texture information as heuristic, because approaches in the past rely on features that are largely lost after image corruption and are not generalizable to different GANs. However, our theoretical reasoning and empirical studies both lead to the realization that their GLCM descriptor, as a global texture descriptor, is still not robust enough in all cases of image manipulation, because it doesn't take enough pixels into account, making it distortion-prone. Statistical analyses show that LBP is more generalizable to different GANs, and also reaches high consistency in outputs after image manipulation. Motivated by this finding, we implemented a Convolutional Neural Network with a ResNet backbone that uses LBP to enhance its global texture perception, effectively describing the texture at various semantic levels in an image with improved robustness. We conducted experiments with our model on images generated from StyleGAN and StyleGAN2, as well as images manipulated by different filters, and showed that our model reaches better and more consistent performance during image manipulation or in cross-domain settings, especially when images are subjected to Gaussian noise, in which we reached a performance increase from 82% *to* 90%*. We open-sourced our code at* `https://github.com/Josh-Cena/lbpnet`.

Keywords: *Texture descriptor, Local binary pattern, Fake face detection*

1 Introduction

With the rapid development in Generative Adversarial Networks (GANs) [12, 3, 13, 14], fake face generation has been a field of increasing attention in recent years. The images generated are of such high quality that no human observer could possibly discriminate them from real faces. In Figure 1, three out of the six are generated using the latest generative model, StyleGAN2 [14], while the other three are real. Can you tell the fake from the real?[1] Because computer-synthesized fake faces are so highly

[1]Answer: the 1, 3, 6 faces are fake

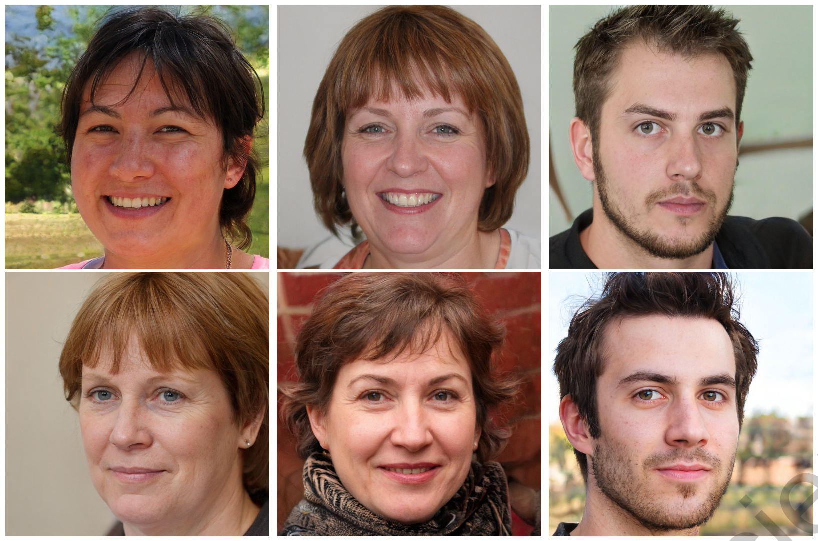


Figure 1: Three of these faces are generated with StyleGAN2 [14] and three are real people's photos. Can you tell which ones they are?

deceiving, political and social concerns naturally arise, such as forged faces being used to bypass face recognition checks, generate prank photos, and spread false information on the Internet. This weakens the credibility of mass media, exposes security vulnerabilities, and creates social unrest.

As human efforts in detecting fake faces have mostly been in vain (proven empirically by the example above), a fake-face classifying neural network becomes an urgent need. Existing algorithms make use of different information, including color distribution [2], reflection spectrum [10, 15, 17], and texture fingerprints [4, 21, 19] to classify images as "real" or "fake", but many of them have severely degraded performance when images have been manipulated, or when the network is trained and tested using images from different GANs, due to the information they rely on not being robust enough. As image manipulations and difference among generative models significantly distort those cues that neural networks like ResNet [9] and Co-detect pick up, it becomes vital to direct the neural network to focus on more robust features that are inherent to the target.

A recent work by Liu *et al.*, named GramNet [19], proposed using global textures as a robust feature to address the cross-GAN and image manipulation issues, and conducted a series of empirical studies to support their hypothesis. Global textures are persisted after image manipulation and tend to be more consistent among GANs than between fake and real. They used the handcrafted Gray-level co-occurrence matrix (GLCM) [8] to describe global texture, but still, their own experiments suggested that the GLCM as a heuristic is not able to describe the texture robustly in every case, particularly when the image is passed through a blur or noise filter. For more details, see Section 3.1.

In this paper, we propose a new architecture, named LBPNet, which inserts the local binary pattern (LBP) [24] as texture descriptor into image classification networks to achieve high manipulation robustness in all scenarios. We conducted variable-controlled empirical studies with LBP and GLCM as texture descriptors on fake face image datasets, and discovered that while larger-scale textures are preserved almost perfectly even after severe image degradation, in all reception field sizes, LBP-described textures are more resilient than GLCM-described ones. LBP also has more output dimensions under similar setup, leading to richer information contained within the descriptor.

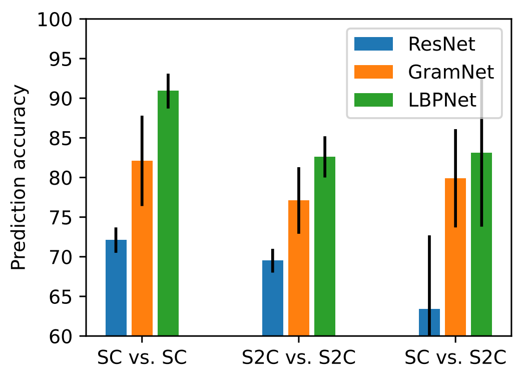


Figure 2: Our LBPNet has a significant performance boost when analyzing noise-filtered images, and stays generalizable when the testing and training set are from different GANs. SC and S2C are two datasets—StyleGAN-CelebA and StyleGAN2-CelebA. "SC vs. S2C" is a cross-domain setting.

Our contributions are summarized as follows:

- We conducted theoretical and empirical studies to reveal that the GLCM applied to the original GramNet [19] cannot satisfactorily handle manipulations in different domains (i.e., blur, noise, and down-sampling between the same or different GANs). Compared with GLCM, in the same domain (StyleGAN [13]), the consistency of LBP global texture description is improved by 4%, 6%, and 8% in the correlation of blur, noise, and downsampling, reaching 98%, 96%, and 98%, respectively. In cross-data domains (StyleGAN [13] and StyleGAN2 [14]), the performance of LBP in blurring and down-sampling is comparable to that of GLCM, but the noise has an absolute improvement of 4%.
- We designed our LBPNet architecture by coupling the LBP descriptor with the classic ResNet architecture, making the image classifier focus more on the texture information described by the local binary pattern. The backbone of ResNet is used as a feature extractor, where each convolutional layer generalizes the image to a higher semantic level. Meanwhile, an LBP analyzer is applied to the output of each convolutional layer to describe their global texture. We open sourced our code on GitHub: `https://github.com/Josh-Cena/lbpnet`.
- We reproduced GramNet proposed in the previous work [19], and crafted a $5k$-image dataset of faces of mixed identities (real or fake) by training StyleGAN and StyleGAN2, following their experiment setup. The dataset is further diversified by manipulating the images with various image filters. The complete code to generate the dataset is also open sourced along with the model.
- We conducted in-domain and cross-domain training and testing on LBPNet and GramNet. The experiment demonstrates that the performance of LBPNet is more manipulation invariant and complements the shortcomings of GramNet, as shown in Figure 2. Especially in Gaussian noise, the prediction accuracy is improved by 6%, while in other manipulations, it is the same as GramNet.

2 Related work

In this section, we will briefly introduce the progress in related fields. First, we will explore modern image generating models available in the wild. Then, we will introduce the principal methods for identifying generated faces, comparing them with our approach. Lastly, we will enumerate a few examples of LBP and other texture descriptors in computer vision, and explain why LBP has the potential to yield promising results.

2.1 GAN-generated fake faces

Goodfellow was the first to coin the term Generative Adversarial Networks (GANs) [7], describing a GAN as "a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake". Further work has been done to apply GANs to areas including text generation, image generation, game AI, etc.

Specifically, in the field of fake-face-generation, the most widely used GANs are the StyleGAN family, developed by the team of T. Karras. The first generation, Progressive Growing of GANs (PGGAN) [12], reached state-of-the-art performance in both variation and quality with three innovations: (1) progressive training with inputs of improving quality; (2) using minibatch standard deviation; (3) normalization in the learning rate and feature vector of generator and discriminator. The group went on to develop *StyleGAN* [13], which took an alternative approach of using style-based generation rather than directly providing the generator with the latent code. The image quality was further improved by removing normalization artifacts, enhancing image smoothness, and correcting eyes and teeth that do not have natural positions, the resulting architecture known as *StyleGAN2* [14].

Besides StyleGAN, popular models include StarGAN [3], which trains a single model across multiple datasets and domains. It is targeted towards *partial* face forgery, e.g., changing facial expressions or attributes, so it won't be the main topic of this paper, which focuses on entire face synthesis.

2.2 Fake face detection

Based on our survey, we divided fake face detection into two categories: leveraging texture information and semantic information. Semantic information refers to any feature that may be picked up by human observers, e.g., earrings, letters, illumination, etc., while texture information often relies on computer analysis of pixel values and is almost completely ignored by human discriminators.

Semantic analysis often relies on understanding of how objects appear in real life. S. Hu *et al.* analyzed corneal highlight to detect inconsistency in the reflection spectrum [10]. Similarly, the reflectance intensity of skin regions under different wavelengths can be used to discriminate between "natural lighting" and "synthesized lighting" [15, 17]. These solutions are more viable than detecting objects, because the latter do not appear consistently in every image. Corneal highlight is also unlikely to be disguised, compared to mouths and noses, which may be masked (especially in the current pandemic). However, semantic information are prone to corruption as JPEG compression or color distortion during digital transmission easily hides away small-scale details, which limits their usage.

Texture information yields more corruption robustness due to its ability to capture large-scale patterns. Liu *et al.* proposed the *GramNet* architecture to detect periodic patterns in face images on various semantic levels [19]. They used the gray level co-occurence matrix (GLCM) to describe the texture and drew the conclusion that global textures are more corruption-robust. It is also proven that each GAN leaves its unique "fingerprint" in generated images, which could be extracted and identified [21]. However, this approach fails when the GAN producing the image is not included in the training set, and reliable detection relies on knowledge of all GANs in the wild, a seemingly unviable task.

Apart from gray-scale texture information, color distribution information has also been scrutinized [22, 2]. However, CNNs are observed to reach similar performance when color is removed [19], implying that color distribution may not be the key influencer in fake face detection networks. By a similar argument, Mummadi *et al.* [23] questioned the hypothesis that correcting the strong texture bias in CNNs by improving shape bias can lead to improved performance [6], because the approach of stylization merely increases data augmentation and variable-controlled experiments yield similar performance in non-shape-bias-enhanced images.

An attention map was proposed to improve the face detection algorithm's modularity and explainability [4]. Cao *et al.* utilized this attention-based approach to detect fake faces, achieving promising performance within compressed images by pairing each high-quality input with a low-quality one and extracting compression-insensitive features [1]. This approach is analogous to ours because we also attempt to extract compression-invariant features, yet our use of texture descriptors yields extra explainability and doesn't require knowledge of any high-quality input in training phase.

2.3 Texture descriptors in face detection

There are numerous ways to describe "texture" contained in a image, a lot of which already picked up by deep learning algorithms in face recognition. *Local Binary Pattern (LBP)* [24] and *Histogram of Gradients (HOG)* [20] are two such descriptors. This section is a compilation of miscellaneous investigations in how LBP and HOG can be used in computer vision.

A. Dosovitskiy and T. Brox conducted an interesting experiment to reverse-deduce the original image given a texture descriptor (LBP and HOG for two such examples) [5]. The promising results implied that these descriptors carry much richer information than expected.

Kabbai *et al.* used LBP in place of SIFT in image matching algorithms to achieve greater robustness towards corruption [11]. They demonstrated that after down-sampling or adding noise, LBP is still able to capture texture feature, while SIFT, which relies on intricate edge information, loses grasp. This means LBP and the similar HOG would perform better under image corruption to describe the global texure. LBP and HOG have been used jointly to indentify faces under infrared [25]. Yassin *et al.* did an extensive comparative study using LBP, HOG, SIFT, SURF, etc. in face recognition [16]. VLC and LBP have demonstrated superior performance by reaching the highest true positive rate at every false positive rate. This enhanced the confidence in using LBP and HOG for fake face detection as they will be able to accurately capture human face features.

In the area of fake face detection, Liu *et al.* first incorporated canonical texture descriptors [19]. They used GLCM and contrast coefficient to describe the local- and global-textures, and by plotting the correlation against the "reception field size" of the GLCM, argued that global textures are better preserved after compression. Their final architecture used the Gram matrix, a variant of the covariance matrix, to describe the global texture. We will demonstrate how the direct use of texture descriptors in the network architecture improves explainability and performance.

3 Our method

Liu *et al.*'s proposal of the GramNet [19] is very promising, but even from a theoretical perspective, its performance is degraded when the image is manipulated by a blur or noise filter, which can also be observed from their own experiments. We propose to replace the core texture extractor, Gram Block, with LBP to enhance its capability of recognizing global textures during manipulation. In this section, we will use empirical studies to illustrate how LBP can be more capable of describing textures, and propose our own architecture.

3.1 Pitfalls of using GLCM to describe global texture

Liu *et al.* used a Gray Level Co-occurrence Matrix (GLCM) [8] to describe the global texture. GLCM, as its name (a co-occurrence matrix) suggests, measures how often two pixel values co-exist. Given a pair distance r, it outputs a matrix G of size 256×256, where G_{ij} is the number of occurrences of a pixel with value i which is r pixels to the left, right, above, or below a pixel with value j. To scalarize this matrix, we use a weighted sum of every cell:

$$C = \sum_{i,j=0}^{256} |i-j|^2 G_{ij} \tag{1}$$

which is called the "contrast ratio" of the image.

To verify that global textures are preserved after image manipulation, they computed the contrast ratio for each image before and after processing. As the "reception field size" increases, the Pearson correlation between the contrast ratios also increases, indicating a stronger retainment of texture information.

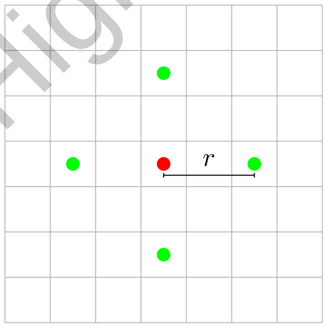


Figure 3: GLCM takes at most four points into consideration when describing textures.

However, GLCM only takes at most 4 points within distance r into account, which is far from being descriptive of the *texture*, especially global texture. It is evident from their own experiment results that the handcrafted GLCM descriptor has degraded performance when applied to noise-filtered images (see Figure 4), which is because of the limited descriptiveness of GLCM. It is very likely that the "co-occurrence" matrix will be significantly distorted when every pixel is shifted randomly and independently by the noise mask.

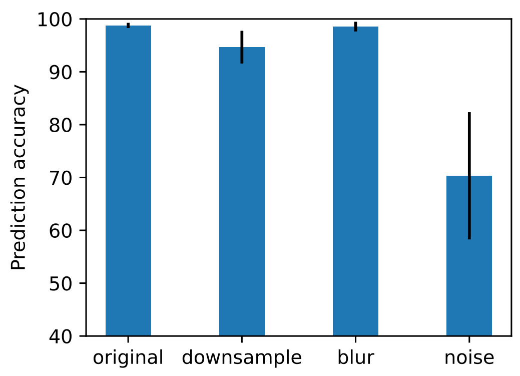


Figure 4: The performance of GramNet, reproduced from the PGGAN vs. CelebA-HQ experiment group in Table 3 of Liu *et al.*'s original paper [19]. There is a large drop in performance in noise-filtered images, and also a greater error.

3.2 Are global textures described by LBP manipulation invariant?

Because GLCM appears to be non-ideal, we decided to adopt LBP as an alternative descriptor in the image classification network. LBP has been widely used in computer vision to analyze texture information, particularly in face recognition. It takes more pixels into account, and generates descriptive results of the *local binary pattern*, as the name goes, as opposed to a scalar contrast index. Therefore, we believe that leveraging LBP as the heuristic for texture recognition can be more manipulation-robust and achieve better results. We conducted some empirical analysis to verify that LBP is truly manipulation-robust in describing global textures.

3.2.1 Mechanism of LBP

Local Binary Patterns [24] is a texture descriptor that summarizes the occurrences of different binary patterns of a given size. It takes two parameters: the reception field radius r and the number of points n.

First, the image is divided into cells of size 16×16. For each pixel x in the cell, the pixel values of the n pixels within radius r from x are compared to x, in a given order (typically clockwise). We index these pixels in that order as y_1 to y_n. Figure 5 gives a rough illustration of how pixels are selected.

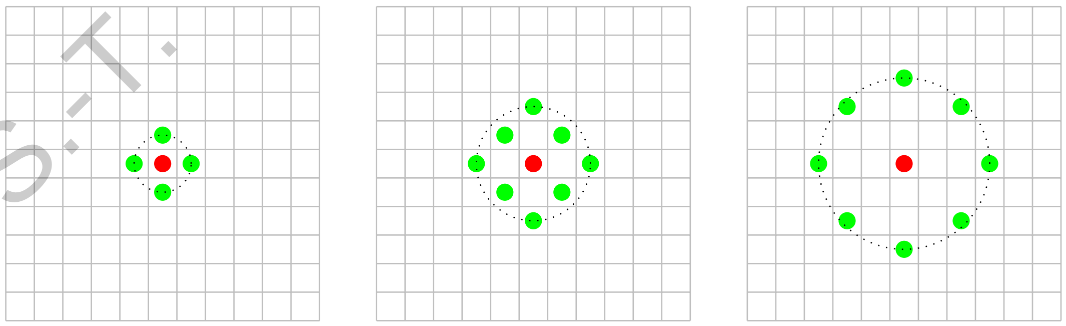


Figure 5: How pixels are selected based on the central pixel in our LBP algorithm. The pair distance r varies from 1 to 3; n is 4 in the first one and 8 otherwise.

For each pixel, the comparison result will be a binary number d with length n describing the “pattern”. Mathematically, this distribution descriptor is calculated as

$$d = \sum_{i=1}^{n} s(y_i, x) \cdot 2^n,$$
$$s(y_i, x) = \begin{cases} 1, & \text{if } y_i \geq x \\ 0, & \text{if } y_i < x \end{cases} \tag{2}$$

Each pixel has a value of d associated, which can be understood as its own “local binary pattern”. The transformation from the original image to the local binary patterns is exemplified in Figure 6.

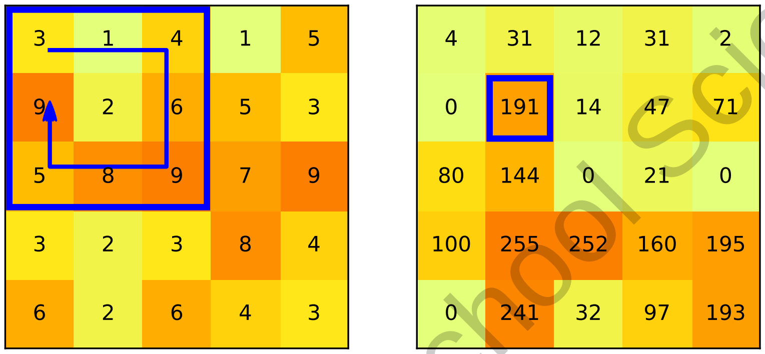


Figure 6: A toy example demonstrating how an image is tranformed to an LBP matrix. The radius r is 2 and the number of points n is 8. the pixel with value 2, when comparing values to its 8 neighbors in the arrow’s direction, yields the pattern 10111111, which becomes 191.

To turn the pattern distribution into a feature vector, we traverse the image and compute the histogram of d, i.e., how often does each pattern occur, yielding a histogram vector of dimension 2^n, corresponding to all possible values of d. In the last step, the histograms from each cell are concatenated.

The LBP algorithm is known for its transformation invariability, and compared to the covariance matrix, LBP can also be variable in its reception field size. From its principle, we also reason that it is more manipulation-robust, because it takes more pixels into account when describing texture patterns. This means that corruption to a few pixels or local sites will not distort large-scale textures as other non-corrupted points balance out the fluctuations. To further prove our point, we conducted a statistical analysis on our dataset using LBP to verify that it is manipulation-invariant.

3.2.2 Statistical analysis of LBP’s performance at capturing global texture

We followed a similar experiment setup as Liu *et al.* [19]. The dataset generation method is discussed in Section 4.1. We used the LBP implementation from `skimage.feature`[2], with radius $r \in \{1, 2, 5, 10, 15, 20\}$, and number of points $n = \max\{8r, 16\}$. In other words, except for $r = 1$ where

[2] https://scikit-image.org/docs/dev/api/skimage.feature.html#skimage.feature.local_binary_pattern

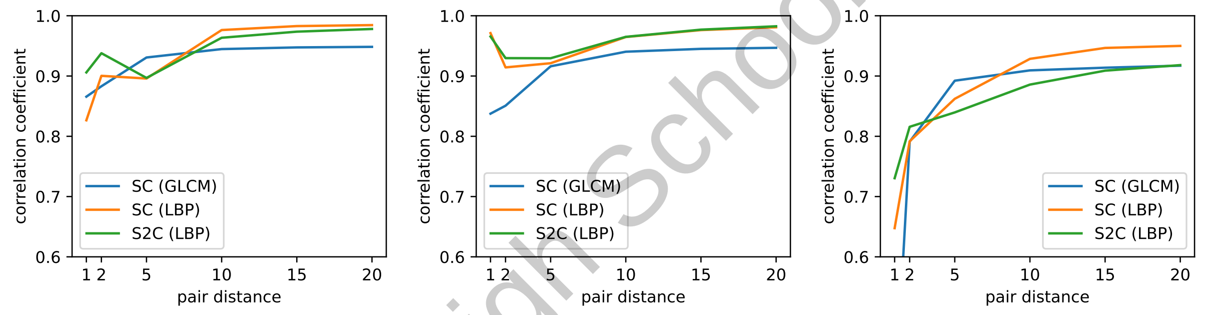
$n = 8$, n is equal to 16 for all the other radii. The output of the LBP algorithm is a vector with length 2^n.

To verify that global textures are more robust over long distances, we would increase the radius of our "reception field" and find whether how correlated the textures are before and after modification. For each image $\mathbf{I}$ and their corresponding processed image $F(\mathbf{I})$, we computed their LBP vectors $\mathbf{X} = LBP(\mathbf{I}), \mathbf{Y} = LBP(F(\mathbf{I}))$, and found the pearson correlation coefficient $\rho_{\mathbf{X},\mathbf{Y}}$. We took the average of all images, and use this value $\overline{\rho}$ as the "correlation factor" at this distance.

Pearson correlation coefficient is a measure of how linearly related two sets of values are. Given two one-dimensional vectors $\mathbf{X}$ and $\mathbf{Y}$, the correlation coefficient is given by

$$\rho = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}. \tag{3}$$

The closer r is to 1, the more linearly correlated the two data vectors are. Therefore, the prediction from our hypothesis is that as the pair distance in the LBP algorithm increases, the Pearson correlation coefficient also increases. We used the implementation of Pearson correlation from `numpy.corrcoef`[3].



(a) Down-sample (b) Blur (c) Noise

Figure 7: Pearson correlation of texture feature vectors between original images and their manipulated counterparts. There is generally more correlation as the pair distance increases, i.e., when the feature becomes more global, while LBP shows better consistency on larger scales.

The complete results are shown in Figure 7. The correlation factor generally displays an increasing trend when the reception field radius increases from 5 to 20, and reaches a correlation of $\rho > 0.95$ for all modification methods when $r = 20$. Interestingly, the behavior is less deterministic when $r < 5$, because the correlation *decreases* in both (a) Down-sample and (b) Blur. We reason that this is because there is a behavior inconsistency when r increases from 1 to 2: more points are taken when $r = 2$. We couldn't have taken 16 points when $r = 1$, however, because there are just 8 points with distance 1 from the central pixel. Nevertheless, the global optimum for all curves occur at $r = 20$, verifying our hypothesis that the LBP is able to capture global texture information after image modification in the same way as Gram matrix.

It is evident from Figure 7 that LBP is more manipulation-invariant at larger scales, achieving about 5% increase in correlation compared to GLCM. Moreover, it also shows good generalizing abilities,

[3] https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html

because the curves from SC (StyleGAN-CelebA) and S2C (StyleGAN2-CelebA) have nearly the same values everywhere.

The superiority of LBP is, from our description of its mechanism, that it is actually a description of *patterns* of a given size, rather than merely counting the co-occurrences between discrete points. The latter can be subject to great fluctuations when pixels are randomly adjusted by the noise filter.

3.3 Architecture of LBPNet

To directly compare the effectiveness of LBP to the baseline model, GramNet, we designed our model with the same ResNet-18 [9] backbone, substituting the Gram block with an LBP block. The outputs from each convolution layer are concatenated to obtain a feature vector, which is then passed through a fully connected layer. The architecture is illustrated in Figure 8. In the next subsections, we will describe in detail the designs of the backbone and our LBP blocks.

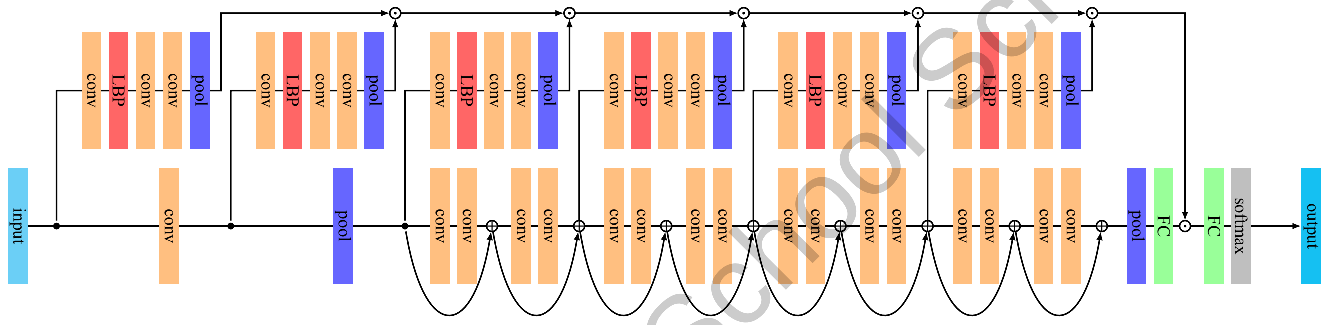


Figure 8: Our LBPNet architecture. $\odot$ represents vector concatenation, while $\oplus$ represents vector addition. The input is a gray-scale raw image, and output is the biclassification result from Softmax ("real" or "fake"). The main backbone (on the bottom) has the exact same dimension settings as the original ResNet-18 [9], and the LBP blocks have similar design as GramNet [19].

3.3.1 ResNet backbone

The LBPNet architecture uses ResNet-18 as the backbone. Like other Deep convolutional neural networks, ResNet is primarily composed of a series of convolution layers which extract feature maps on different semantic levels. ResNet inserts shortcut connections between every convolution layer, which permits residual functions to be passed through the layers and thus mitigates the problem of degradation as the convolution depth stacks up and the residual approaches zero.

We could view each layer in the ResNet as a feature extractor, which turns an image into a feature map, or a lower-level feature map into a feature map of features. This permits the network to continuously recognize semantic information of larger and larger scale. This permits us to integrate our LBP algorithm directly on top of the convolution output: the LBP will describe the distribution of features extracted on different semantic levels.

ResNet-18 is the model from the set of PyTorch implementations[4] with the smallest number of convolution layers. We selected this model because as we are attaching an LBP block to every layer, we

[4]https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py

Table 1: The dimensions for each LBP analyzer block's input and output.

Level	Input dimensions	LBP radius r	Output dimensions
1	$m \times n \times 1$	1	$\frac{m \cdot n}{16} \times 1 \times 32$
2	$m \times n \times 64$	1	$\frac{m \cdot n}{16} \times 1 \times 32$
3	$m \times n \times 64$	2	$(m \cdot n) \times 1 \times 32$
4	$m \times n \times 128$	4	$(m \cdot n) \times 1 \times 32$
5	$m \times n \times 256$	8	$(m \cdot n) \times 1 \times 32$
6	$m \times n \times 512$	8	$(m \cdot n) \times 1 \times 32$

need to restrict the output dimensions to a reasonable size. However, the same design principle can be applied to deeper convolutional networks, just by distributing the insertion of LBP blocks more sparsely (e.g. one block every four convolutional layers).

3.3.2 LBP analyzer block

We insert our LBP analyzer block at six levels of convolution. In each block, the input image is first convolved by 32 convolutional map layers $\mathbf{b}^l, 0 \leq l < 32$. The outputs are passed through the LBP layer which can be described by:

$$\mathbf{L}_{ij} = \sum_{i=0}^{7} s(\mathbf{B}_i * \mathbf{I}) \cdot \mathbf{v}, \tag{4}$$

where $\mathbf{I}$ is the input image with dimensions enlisted in Table 1, $\mathbf{B}_i$ is the convolutional filter with radius r, and $\mathbf{v}_i = 2^i$ are the weights. Equation 4 is equivalent to Equation 2, re-written in vector form to reveal its nature as a convolution process. The radius r increases as the image passes through each convolution layer, because the feature maps are becoming increasingly "global", hence the reception field sizes of the LBP layers also increase.

The output from the LBP, before being transformed into a histogram (which is, essentially, a pooling layer), has the same dimensions as the original input. After pooling, the LBP is vectorized so that outputs from different levels can be concatenated.

$$\mathbf{F} = \mathbf{L}'_0 \oplus \mathbf{L}'_1 \oplus \cdots \oplus \mathbf{L}'_5 \tag{5}$$

where $\mathbf{L}'_i$ is the vectorized feature map from the i-th LBP analyzer block. $\mathbf{F}$ is the final feature vector, which, after the final fully connected layer and softmax layer, yields the prediction result.

In our model, the feature vectors are concatenated across the layers to obtain a final result. The rationales behind this are:

1. Each component in the feature vector can be viewed as a unique pattern contained in the image. All the components are of equal importance, and by concatenating them, we obtain a fair representation of the holistic feature distribution within the image.

2. A similar architecture has been proposed by Liu *et al.* [19], and through this similar setup, cross-model performance comparisons can be conducted fairly. It will not be hard to improve the integration method, or use an alternative backbone other than ResNet-18.

4 Experiment

4.1 Crafting dataset

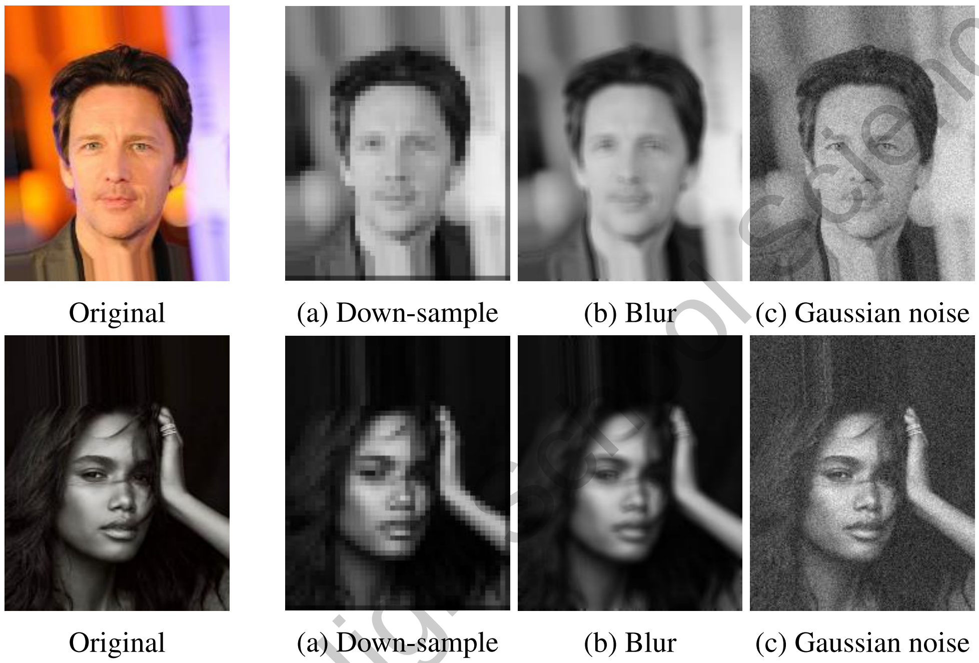


Figure 9: Results of different image manipulation techniques applied to the CelebA dataset. Note that some images from this dataset (such as the second here) are already in gray-scale, hence no conversion is needed.

We first acquired a dataset of original and processed faces, which we will also use for model training. Following the previous work [19], we chose CelebA [18] as the dataset for real faces. It contains 200,000+ face images, each labeled with style attributes, but which we will not pick up. We utilized three common image modification techniques to simulate image corruption in the real world: down-sampling, blur, gaussian noise.

- *Down-sampling*. Block size is set to $(4, 4)$, and the block reducer is an arithmetic mean, which produces a quite convincing down-sampled image. We used the implementation of `block_reduce`[5] from `skimage.measure`.

[5]https://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.block_reduce

- *Blur*. We used the implementation of `PIL.ImageFilter`[6], with the defalt setup of a convolution kernel with size 5×5:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{6}$$

- *Gaussian noise*. This is achieved by generating a normal distribution mask with $\mu = 0, \sigma = 16$, applying to the original image, and then normalizing pixel values to $[0, 255]$. In mathematical terms, we first compute the mask

$$\mathbf{M}_{ij} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right), \sigma = 16 \tag{7}$$

  And then apply this mask to the original image with normalization:

$$\begin{aligned} \mathbf{I}' &= \mathbf{I} + \mathbf{M} \\ \mathbf{I}''_{ij} &= \frac{255}{\max(\mathbf{I}') - \min(\mathbf{I}')} \left(\mathbf{I}'_{ij} - \min(\mathbf{I}')\right) \end{aligned} \tag{8}$$

We randomly selected 20,000 images from CelebA, converted them to gray-scale, and applied the filters above. The results of manipulation for two images are shown as examples in Figure 9. To make manipulation more convenient, all images are pre-processed with the gray-scale filter. This step is justified by numerous research papers demonstrating that CNNs are more focused on texture information than color information.

Next, we trained a StyleGAN[7] and a StyleGAN2[8] model on CelebA with pre-trained parameters, and acquired two fake face datasets: *StyleGAN-CelebA* and *StyleGAN2-CelebA*, each with size of about 3000. They will be used to conduct in-domain and cross-domain experiments. We also applied the image filters discussed above, i.e., down-sample, blur, noise, to these fake faces.

Table 2: Performance of different LBPNet compared to baseline models with in-domain settings, tested with images processed with different manipulations.

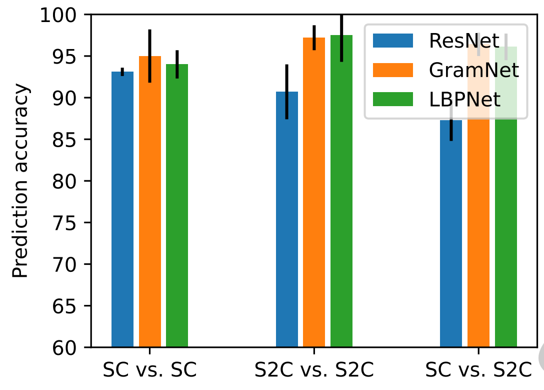
Dataset	Model	Original	Down-sample	Blur	Noise
StyleGAN-CelebA [13]	ResNet [9]	93.1 ± 0.5	80.7 ± 1.2	75.2 ± 0.9	72.1 ± 1.6
	GramNet [19]	$\mathbf{95.0 \pm 3.2}$	90.7 ± 2.8	92.6 ± 1.9	82.1 ± 5.7
	LBPNet	94.0 ± 1.7	$\mathbf{91.3 \pm 1.9}$	$\mathbf{93.0 \pm 1.4}$	$\mathbf{90.9 \pm 2.2}$
StyleGAN2-CelebA [14]	ResNet [9]	90.7 ± 3.3	76.0 ± 1.0	78.0 ± 2.1	69.5 ± 1.5
	GramNet [19]	97.2 ± 1.5	89.0 ± 1.1	$\mathbf{90.6 \pm 1.0}$	77.1 ± 4.2
	LBPNet	$\mathbf{97.5 \pm 3.2}$	$\mathbf{90.4 \pm 2.1}$	89.1 ± 1.8	$\mathbf{82.6 \pm 2.6}$

[6] https://pillow.readthedocs.io/en/stable/reference/ImageFilter.html
[7] https://github.com/NVlabs/stylegan
[8] https://github.com/NVlabs/stylegan2

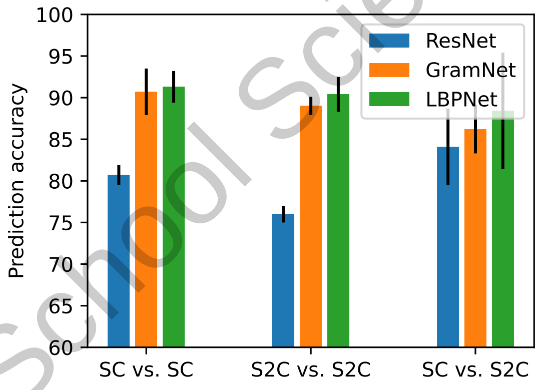
4.2 Implementation and setup

Our model is implemented with PyTorch. The datasets are the ones collected as aforementioned, with $1k$ real faces and $1k$ fake faces for training. For each model, the learning rate is fixed at 10^{-5}. We trained for a maximum of 10 epochs, validating with a 500-image holdout set of mixed identities (original or manipulated) at the end of each epoch. The models are then tested using another test set consisting of $1k$ faces.

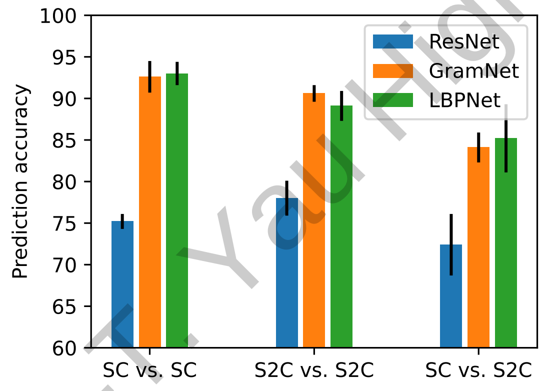
The experiment consists of three sets: two in-domain experiments using StyleGAN-CelebA (SC) and StyleGAN2-CelebA (S2C) as both training and testing set, and one cross-domain experiment using StyleGAN-CelebA for training and StyleGAN2-CelebA for testing. The baseline model is ResNet-18, the backbone model. In addition, we implemented our own version of GramNet [19] to compare out performance to. Image sizes are all rescaled to 256×256, the size that the original CelebA dataset uses.



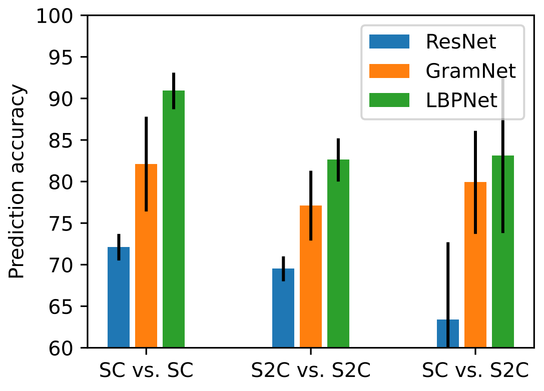
(a) Original images



(b) Down-sampled images



(c) Blurred images



(d) Noise-filtered images

Figure 10: The performance of the three CNN models: ResNet, GramNet, and LBPNet, on different datasets. SC stands for StyleGAN-CelebA, and S2C, StyleGAN2-CelebA.

4.3 Experiment results

All the experimental results are summarized in Table 2 and Table 3. Figure 10 provides visualization of all data combined.

Table 3: Performance of LBPNet compared to baseline models with cross-domain settings.

Training set	Testing set	Model	Original	Down-sample	Blur	Noise
StyleGAN-CelebA	StyleGAN2-CelebA	ResNet [9]	87.3 ± 2.5	84.1 ± 4.6	72.4 ± 3.7	63.4 ± 9.3
		GramNet [19]	$\mathbf{96.4 \pm 1.4}$	86.2 ± 2.9	84.1 ± 1.8	79.9 ± 6.2
		LBPNet	96.1 ± 1.6	$\mathbf{88.4 \pm 7.0}$	$\mathbf{85.2 \pm 4.1}$	$\mathbf{83.1 \pm 9.3}$

The error values mostly stay at around 2.0, signaling a comparatively consistent performance of all models. However, LBPNet experiences a significantly higher fluctuation in prediction accuracy in the cross-domain setting. The same behavior was observed in the GramNet paper [19] as well, where the error once reached 12.0 (Table 3, PGGAN - PGGAN, Noise), and is generally higher than our experiments by about 5%. We reason that this is because of the unique texture fingerprints inherent to each GAN [21], described in Section 2.3. Liu *et al.* used PGGAN as their cross-domain model, while we adopted the novel StyleGAN2. Because StyleGAN and StyleGAN2 operate with similar principles (style-based generation), the texture fingerprints are also likely to be more similar than between PGGAN and StyleGAN. Based on this hypothesis, we predict that if the cross-domain experiment is extended to PGGAN or other non-style-based GANs, we should observe a larger fluctuation in performance than currently.

LBPNet improves in most scenarios of image processing compared to GramNet (8 out of 12 experiment setups), while consistently outperforming the ResNet baseline. LBPNet has a most significant improvement in the case of Gaussian noise, reaching at most ten-percent increase in prediction accuracy. There is a slight regression in one setting involving blurred images and two with original images. We reason that these are the cases in which GLCM is actually behaving properly as a texture descriptor, because the *co-occurrence* pattern is not distorted by manipulation. Gaussian blur is a uniform manipulation applied to every pixel through a convolution kernel, and thus the GLCM texture is preserved. Contrastly, down-sampling involves information loss, and Gaussian noise is non-uniform. In these cases, LBP extracts textures more robustly than GLCM, thus bringing a performance boost.

Compared to Liu *et al.*'s reported results (one set of experiments is shown in Figure 4), our reproduced results are generally worse. This could be due to a lack of training data, because their training used a more diverse ($20k$) dataset, with higher resolution (512×512). However, due to the computing power and resource limitations, we were unable to fully replicate their setup. Still, under the same dataset and computing conditions, LBPNet has a better performance.

5 Conclusion

In this paper, inspired by the approach of using global textures in fake face detection [19], we explored the possibility of incorporating LBP as a texture extractor in image classification networks like ResNet [9]. Our statistical studies show that LBP carries rich texture information that is robustly preserved in all cases of image manipulation, and is also generalizable to cross-GAN settings. Subsequently, we incorporated LBP into the ResNet-18 backbone as an LBP analyzer block, and used it to extract textures on various semantic levels from the convolution output. The model's performance in manipulations such as Gaussian noise and down-sampling shows a consistent improvement compared to the past research using GLCM as descriptor. In cross-domain settings, LBPNet is able to generalize global texture information across images generated by different GANs, and also reaches high prediction accuracy.

References

[1] S. Cao, Q. Zou, X. Mao, and Z. Wang. Metric learning for anti-compression facial forgery detection, 2021. 5

[2] S. Chen, T. Yao, Y. Chen, S. Ding, J. Li, and R. Ji. Local relation learning for face forgery detection. Proceedings of the AAAI Conference on Artificial Intelligence, 35(2):1081–1088, May 2021. 2, 5

[3] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. 1, 4

[4] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. K. Jain. On the detection of digital face manipulation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 2020. 2, 5

[5] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 2016. 5

[6] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness, 2019. 5

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014. 4

[8] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. IEEE Transactions on Systems, Man, and Cybernetics, SMC-3(6):610–621, 1973. 2, 6

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016. 2, 10, 13, 15

[10] S. Hu, Y. Li, and S. Lyu. Exposing GAN-generated faces using inconsistent corneal specular highlights, 2020. 2, 4

[11] L. Kabbai, A. Azaza, M. Abdellaoui, and A. Douik. Image matching based on LBP and SIFT descriptor. In 2015 IEEE 12th International Multi-Conference on Systems, Signals Devices (SSD15), pages 1–6, 2015. 5

[12] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation, 2018. 1, 4

[13] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks, 2019. 1, 3, 4, 13

[14] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of StyleGAN, 2020. 1, 2, 3, 4, 13

[15] Y. Kim, J. Na, S. Yoon, and J. Yi. Masked fake face detection using radiance measurements. Journal of the Optical Society of America Association, 26(4):760–766, Apr 2009. 2, 4

[16] Y. Kortli, M. Jridi, A. A. Falou, and M. Atri. A comparative study of CFs, LBP, HOG, SIFT, SURF, and BRIEF techniques for face recognition. In M. S. Alam, editor, Pattern Recognition and Tracking XXIX, volume 10649, pages 184 – 190. International Society for Optics and Photonics, SPIE, 2018. 5

[17] J. Li, Y. Wang, T. Tan, and A. K. Jain. Live face detection based on the analysis of Fourier spectra. In A. K. Jain and N. K. Ratha, editors, Biometric Technology for Human Identification, volume 5404, pages 296 – 303. International Society for Optics and Photonics, SPIE, 2004. 2, 4

[18] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015. 12

[19] Z. Liu, X. Qi, and P. H. Torr. Global texture enhancement for fake face detection in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 2020. 1, 2, 3, 5, 6, 7, 8, 10, 12, 13, 14, 15

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91–110, 11 2004. 5

[21] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi. Do GANs leave artificial fingerprints?, 2018. 2, 5, 15

[22] S. McCloskey and M. Albright. Detecting GAN-generated imagery using color cues, 2018. 5

[23] C. K. Mummadi, R. Subramaniam, R. Hutmacher, J. Vitay, V. Fischer, and J. H. Metzen. Does enhanced shape bias improve neural network robustness to common corruptions?, 2021. 5

[24] T. Ojala, M. Pietikinen, and T. Menp. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7):971–987, 2002. 2, 5, 7

[25] Z. Xie, P. Jiang, and S. Zhang. Fusion of LBP and HOG using multiple kernel learning for infrared face recognition. In 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), pages 81–84, 2017. 5

6 Acknowledgements

My first experience with fake faces came from a news report on https://thispersondoesnotexist.com/, a website that shows faces generated by StyleGAN2. Although I had some exposure to generative models, I never came to realize that they could be so deceptive. Being a debater, I started pondering about the social implications this technology could bring, and thus actively explored the area of fake face detection to search for a countermeasure. I came up with this topic when I read Liu *et al.*'s paper on GramNet, which instantly interested me. I had been reading about image analysis techniques at that time, and things like GLCM, Gaussian blur, etc., struck me as familiar. My instincts told me, however, that there's something non-ideal about their use of GLCM in global texture analysis, so I decided to take an alternative approach of use LBP, another concept I dug up in my exploration. GramNet was especially modular in nature, almost plug'n'play style, so I easily adapted it to my own design.

This project benefited greatly from the courtesy of UCloud, who sponsored us with a free 2-GPU server instance for four months.

My parents were always there to support me and push me forward. I'm especially grateful for their understanding and allowing me to stay up late at night to write my paper (they always advocate for going to bed early, but I'm usually productive after midnight).

My tutor taught me valuable skills like how to read research papers, and helped me formalize my research plan when I presented my idea to him.

I'd like to thank my math teacher, Zhao Zhe, for being extremely lenient on homework submissions. I had two coinciding deadlines, and he generously postponed the math paper submission date as much as possible, so that I could devote my full effort to this paper to make it a polished one.

Several of my friends helped me proof-read my work and make it easily understood. They knew almost nothing about artificial intelligence or computer science in general—but that's exactly what helped.