Data Preprocessing in Fake Image Detection*

Yang Yue
Statistics
yangyue3@illinos.edu

***Abstract*—We compare the performance impact of different preprocessing technique combinations on fake image detection. Our support vector machine model's accuracy is non-monotone with respect to image resolution, and is negatively affected by edge detection. The convolutional neural network favors high resolution image input and dislikes randomness introduced by preprocessing techniques. We conjecture that fake images are defined by pixel value anomalies instead of contour or shape anomalies.**

***Index Terms*—Image Preprocessing, Support Vector Machine, Convolutional Neural Network**

I. Introduction

In this project, we compare the fake image detection performances of different combinations of preprocessing techniques in the context of support vector machine and convolutional neural network. Our research is motivated by the lack of consensus on the best practices for image preparation. It is not uncommon to see practitioners overcome insufficient data preparation with brute force by allocating more computational resources to the training process. While the solution is valid, fine tuning the pipeline to improve efficiency is highly relevant as well. Moreover, with the recent push for model explainability, having a better understanding of how preprocessing techniques affect model performance can highlight elements in the data that are important to classification. This knowledge can help us design more effective training regime.

II. Data Review

The Kaggle data set [1] comes with $100,000$ training examples and $20,000$ testing examples of faces that are labeled as either real or fake. The number of faces in each class are balanced in both the training and the testing set. Some sample data points are shown in Fig. 1.

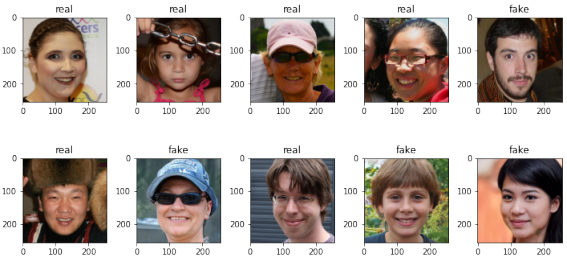

Fig. 1: Sample Images

*The project is prepared for ECE598 Generative AI offered by the University of Illinois Urbana-Champaign.

III. Suppose Vector Machine

There are four preprocessing techniques at maximum applied to an image in our study: resizing, gamma correction, grayscale, and edge detection. Due to the computational cost of support vector machine, resizing is always applied with either 32, 64, or 128 as the output dimension. Grayscale is always applied too due to the same reasoning. We use 3-digit binary sequences to represent the existence of the remaining techniques used in the preprocessing step.

- First digit: 1 if gamma correction is on. 0 otherwise.
- Second digit: 1 if gamma correction is applied to YCbCr image. 0 if gamma correction is applied to RGB image. [1]
- Third digit: 1 if edge detection is applied. 0 otherwise.

The impact of each processing technique applied to the images is visualized in Fig. 2.

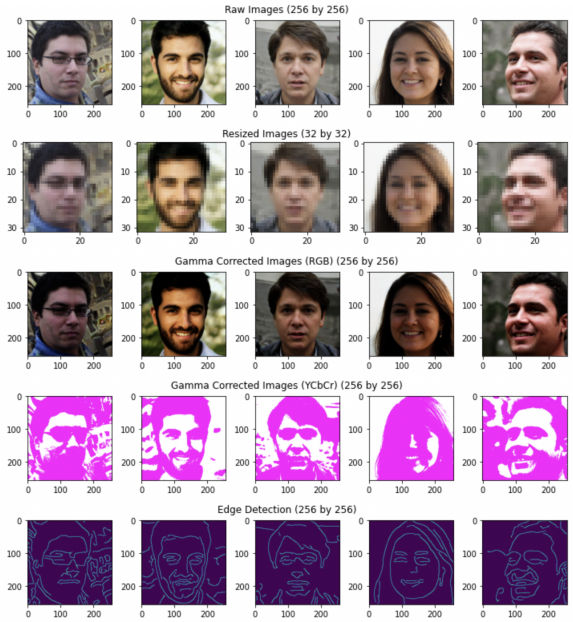

Fig. 2: Visual Impact of Preprocessing Techniques

Resizing is straightforward. It scales the images to a specific shape. As we reduce the dimension, the images become fuzzier.

Gamma defines the relationship between a pixel's numerical value and its actual luminance. With a value of 2, this

[1]We always convert the image back to RGB after applying the gamma correction to the YCbCr image. The second digit is automatically 0 of the first digit is 0.

technique makes the images look like they were taken under different lighting conditions. Our interpretation is that certain features are highlighted with gamma correction.

Applying gamma correction to YCbCr based images then map the color channels back to RGB changes the look of the images drastically. In some cases, contours of some shapes are preserved while others are abstracted away.

Lastly, edge detection greatly simplifies the images. It focuses only on the boundaries of objects by detecting discontinuities in brightness.

The testing accuracy of the support vector machine model with different combinations of preprocessing techniques are summarized in Fig. 3.

	dimension	000	001	100	101	110	111
0	32	0.624	0.534	0.612	0.542	0.620	0.526
1	64	0.508	0.500	0.521	0.490	0.523	0.501
2	128	0.585	0.500	0.591	0.499	0.589	0.509
3	256	NaN	NaN	NaN	NaN	NaN	0.500

Fig. 3: Support Vector Machine Model Summary

Comparing the different resizing dimensions conditional on different processing technique combinations in Fig. 4, we see that the support vector machine algorithm consistently performs the best with the lowest resize dimension among the ones we have experimented. Our initial conjecture is that pixel anomalies that make an image fake can be overwhelmed by regular pixels when the image resolution is high. As a concrete example, suppose the key to detecting a fake face is the pixels around the nose. The rest of the pixels that the image consists of are perfectly normal. Having a 64 by 64 image only serves to dilute the importance of the pixels around the nose, compared with feeding the algorithm a 32 by 32 image. And support vector machine, as an algorithm that seeks to maximize the separation, might be "confused" by the additional redundant information that's not important to the classification.

However, this conjecture is somewhat refuted by the observation that 128 by 128 images leads to a better performance than 64 by 64 images in some settings. This could be a hint of a non-monotone relationship between image resolution and testing performance for support vector machine models. Moreover, it could be the case that the resizing algorithm abstracts away important information under certain dimensions.

We adopt an alternative visualization of the results in Fig. 5. Starting with the 32 by 32 image dimension and comparing the 000 vs. 001, 100 vs. 101, 110 vs. 111 pairs, it's clear that applying edge detection and transform the data from pixel values for boolean values hurts performance. One the one hand, this could be due to imperfect edge detection. As shown in Fig. 2, the chin of the face in the first image isn't properly highlighted. On the other hand, this reinforces the claim that it is primarily pixel value anomalies, not contour/shape anomalies that define a fake image.

Comparing the 100 vs. 110, 101 vs. 111 pairs, it's not immediately clear how applying gamma correction to YCbCr

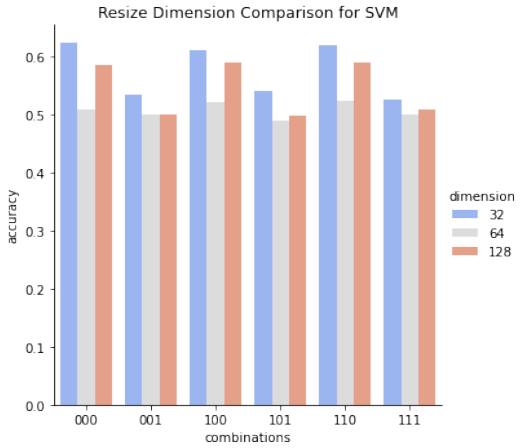

Fig. 4: Resize Dimension Comparison

color channel based images instead of RGB based images affect the model performance.

Comparing the 000 vs. 100, 001 vs. 101 pairs, we cannot draw concrete conclusion regarding how applying gamma correction to RGB images affect performance. Looking at the preprocessing techniques visualizations, we think this lends support to the idea that pixel values anomalies on the contour are key for detection.

The above observations hold true for 128 by 128 resized images. And the negative impact of edge detection on model performance is not as noticeable with 64 by 64 images. As noted before, our support vector machine model performs poorly with 64 by 64 images.

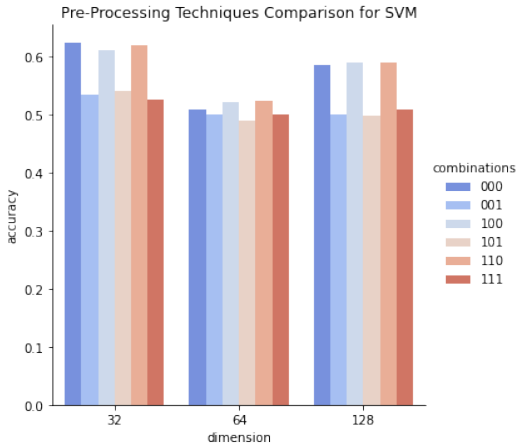

Fig. 5: Preprocessing Techniques Comparison

IV. Convolutional Neural Network

Our convolutional neural network's structure is presented in Fig. 6. Note that our goal is not to create a state-of-the-art network to get the best accuracy, but to compare the impact of using different preprocessing techniques. As long as the model architecture stays constant, we believe our specification does an adequate job.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 63, 63]           6,176
            Conv2d-2          [-1, 64, 30, 30]          32,832
            Conv2d-3          [-1, 64, 26, 26]         102,464
            Linear-4                [-1, 512]      22,151,680
            Linear-5                  [-1, 2]           1,026
================================================================
Total params: 22,294,178
Trainable params: 22,294,178
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.75
Forward/backward pass size (MB): 1.74
Params size (MB): 85.05
Estimated Total Size (MB): 87.54
----------------------------------------------------------------
```

Fig. 6: CNN Model Specification

In this experiment, we also consider four different preprocessing techniques: resizing, color jitter, grayscale, and normalization. Resizing is always on with 64, 128, and 256 as possible output dimension. And just like our support vector machine model based experiment, we use binary sequences to represent the combinations of preprocessing techniques applied to the images.

- First digit: 1 if color jitter is applied. 0 otherwise.
- Second digit: 1 if grayscale is applied. 0 otherwise.
- Third digit: 1 if normalization is applied. 0 otherwise.

The impact of each processing technique applied to the images is visualized in Fig. 7.

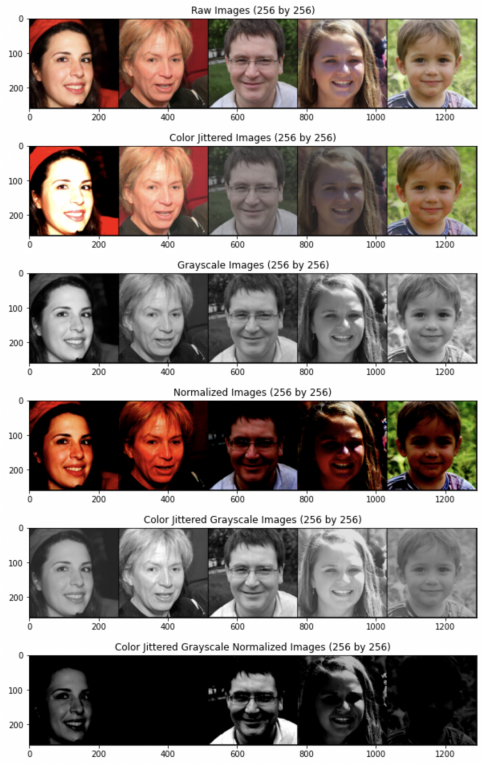

Fig. 7: Visual Impact of Preprocessing Techniques

Color jitter randomly change the brightness, contrast, saturation and hue of an image. To an extent, we are trying to randomly simulate different lighting conditions for the images and see if it improves the network's ability to generalize what it learns in the training.

When grayscale is applied, we choose to preserve three color channels by assigning the same value to them.

Normalization applies feature scaling to the data set. It improves the consistency of the data by mapping the pixel value distributions to ones with the same mean and standard deviation.

The testing accuracy of the convolutional neural network with different combinations of preprocessing techniques are summarized in Fig. 8.

	dimension	000	001	010	100	011	101	110	111
0	64	0.74935	0.80695	0.74840	0.73505	0.79875	0.80465	0.73115	0.71405
1	128	0.80075	0.84105	0.80275	0.73525	0.81925	0.81765	0.78385	0.76505
2	256	0.84155	0.88315	0.87250	0.84060	0.89020	0.89485	0.85350	0.81355

Fig. 8: Convolutional Neural Network Summary

Regardless of the preprocessing technique specifications, higher resolution consistently leads to better detection performance for convolutional neural network, as shown in Fig. 9. This could be another evidence of neural networks being able to pick up and utilize the subtle details in the high resolution images compared to non-deep learning methods.

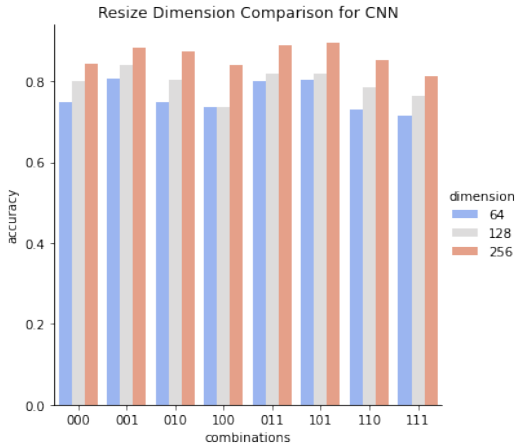

Fig. 9: Resize Dimension Comparison

For a detailed comparative statics analysis, visualized in Fig. 10, we start with the 000 vs. 001, 010 vs. 011, 100 vs. 101, and 110 vs. 111 pairs. It seems that applying normalization leads to performance gain in all pairs except for the last one. As apparent from the visualizations in Fig. 7, applying all three preprocessing techniques make some images unidentifiable. Overall, we still conclude that applying normalization to the images improves the network's ability to generalize the pixel anomalies that define a fake image, and subsequently increases performance.

Looking at the 000 vs. 010, 001 vs. 011, 100 vs. 110, 101 vs. 111 pairs, we get mixed signals. When only grayscale transformation is applied, we get non-negative performance gains that is increasing with image resolution. Conditional on normalization is applied, applying grayscale leads to performance decrease for lower resolution images. Conditional on

applying color jitter, grayscale also increases performance as resolution goes up. Once again, due to the distortion caused by the presence of all three preprocessing techniques, the last pair's comparison doesn't contribute much to the discussion.

Lastly, comparing the 000 vs. 100, 001 vs. 101, 010 vs. 110, 011 vs. 111 pairs, we fail to find an across-the-board consistent observation. Compared with using the null case, applying only the color jitter technique leads to non-positive performance gain. While it could be data set sensitive, perhaps applying random transformation does not help the network to generalize its learning. Conditional on normalization, there's no clear indication how performance is affected. It could be the case that normalization reverts back or weakens some of the changes made by color jittering. Given grayscale images, we see performance decrease for all resolutions. Similar to the observation from applying color jitter to RGB images, our network does not like randomness applied to this data set.

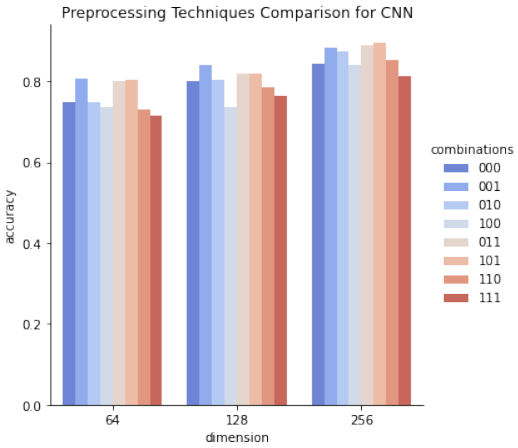

Fig. 10: Preprocessing Techniques Comparison

We also plot the training loss history of our models in Fig. 11 and Fig. 12. For all combinations of preprocessing techniques, using higher resolution images leads to more rapid reduction of training errors. Among the combinations of preprocessing techniques tested, applying nothing but normalization yields the fastest training loss reduction over the epochs. Applying color jitter leads to the slowest training loss reduction. This could be due to the randomness added by color jitter, as noted in previous discussion.

V. Conclusion

We compare the classification performance impacts of different combinations of image preprocessing techniques in the context of support vector machine and convolutional neural network. Our notable findings include

- Support vector machine's performance is non-monotone with respect to input image resolution. Using low resolution images can not only reduces computational cost, but also yield decent performance.
- Edge detection negatively affect support vector machine's performance. Fake images might be defined by pixel value anomalies, not contour or shape anomalies.

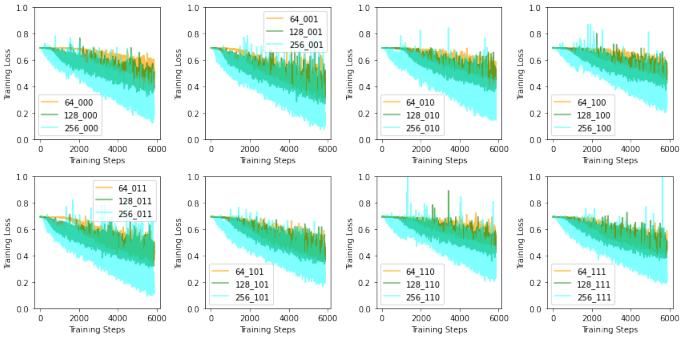

Fig. 11: Training Loss History by Steps

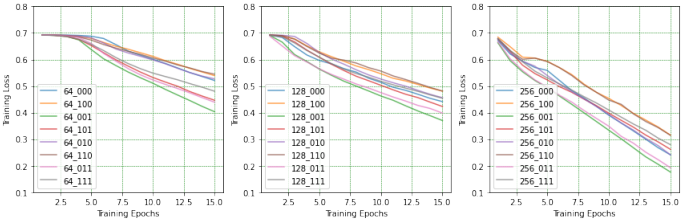

Fig. 12: Training Loss History by Epochs

- Convolutional neural network favors high input image resolution.
- Training data inconsistency due to randomness introduced by preprocessing techniques negatively affect detection performance [2].

Our research sheds some light on the ideal preprocessing practices that could be used for fake image detection. It also contributes to the model explainability literature by highlighting some of the elements in an image that are important to classification, which can help facilitate better model design.

References

[1] X. Lu, "140k real and fake faves", Kaggle .

[2] It's not clear whether this observation is specific to our data set.