Final: Identification of AI-Generated Images

Team 28

Kosta Katergaris
Lindiwe Mutungamiri
Aarti Kalekar
Jereen Valsson

Abstract

Our goal in this project is to improve generalized models that detect if an image is AI-generated or real. Furthermore, now in 2024, AI-generated images are getting better and better, making it harder to distinguish whether an image is AI-generated or real. This could be dangerous because it could be used to generate images with malicious intent. Nevertheless, the contribution to this field our team would like to make is to make a model that detects AI-generated images no matter what model they are made with, and in light of doing so after research, we found a promising method to do so. We hope our team can improve the method that [1] uses by implementing a new state-of-the-art method of attention-based models and modifying the pre-proccessing techniques used by the paper.

1 Methods

1.1 Baseline Approach

After researching various methods, our group found a new method being used by [1], which leverages the texture "fingerprints" left behind by image generative models. Specifically, the inter-pixel correlation contrast between rich and poor texture patches in an image is extracted. This is done by breaking up each image into 192 patches of size 32x32, dividing these patches into two parts (rich texture and poor texture), and then reconstructing them into two 256x256 size images with 64 patches. These patches are measured by their inter-pixel correlation, which is equivalent to their pixel diversity. Furthermore, each image is expected to produce 192 patches; patches would be considered a rich texture patch if they are in the top 33% of inter-pixel correlation values and a poor texture patch if they are in the bottom 33% of inter-pixel correlation values.

1.2 Our Approach

1.2.1 Pre-Processing

The original paper's approach did not highlight how they handled images that were too small, less than 444 X 444 pixels in size, to make 192 patches, each 32 X 32 pixels. In light of this, we found the following resource with its supplemental code [3][9], which is an implementation of [1]. However, they implemented an approach that resizes every image to 256 X 256 and splits the rich and poor texture patches by the variance of pixel correlation values, with rich texture patches being greater than equal inter-pixel correlation variance and poor texture patches being less. Furthermore, they differ from the original approach [1] by averaging all 30 high-pass filters into one image and then passing it into the CNN learnable blocks.

When testing [3]'s approach, we got the same to worse performance than the original model. In response, we developed a method to handle images of all sizes. Instead of resizing every image, we only resized its height and width if they were less than 256 pixels. This will minimize any textural

information lost from resizing the image. Further, we maintained criteria for rich and poor texture patches from [1] for images greater than or equal to 192 patches. For images that produce less than 192 patches, we implemented a similar method duplication approach to [3], using the same splitting criteria. However, after the random patches were made, we sorted them before reconstruction. This is so that we could

Additionally, we used one convolution layer for our high-pass filters that stacked the 30 kernels from [1]'s supplement as frozen parameters with a padding size of 2. This convolution layer has 30 output channels fed to our CNN learnable block, with the same layers as [1]. Note that the convolution layer has 30 input channels, i.e., the 30 feature maps made by the high-pass filters. Furthermore, it has a kernel size of 1 and a padding size of 0 to maintain the original dimensions of the image 256 X 256 and mapped these 30 channels to 3 output channels. This was to maintain the 256 X 256 RGB image structure after our pre-processing technique. We also normalize each photo before being passed to our pre-proccessing.

1.2.2 Classifier

We researched various methods that involved attention-based models. However, we opted to use a SWIN-Transformer v2 as our classifier. This is because it merges well with the pre-processing technique since SWIN-Transformers naturally turn images into patches and apply attention to each window they make. This would help the model better understand the nuances in inter-pixel correlation contrast of AI-generated images vs. real images. We used a pre-trained transformer [10] with a patch size of 4, a window size of 8, and an input size of 256. Our learning rate is 0.00001.

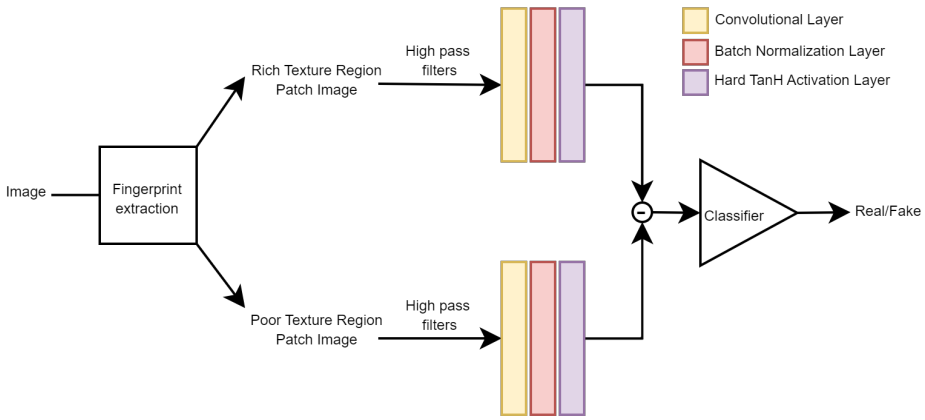

Figure 1: The framework of our approach

1.2.3 Training

For training, we used a batch size of 32 and a learning rate of During testing, we noticed that our validation scores were consistently higher than our training scores, which is peculiar. This was concerning because we thought that our model had a data leak. However, we confirmed that there were no data leaks in our model.

1.2.4 Dataset

The dataset that we chose to use is the GenImage dataset [6]. The GenImage dataset has 2,681,167 images, 1,331,167 of which are real and 1,350,000 fake. Furthermore, there are 1000 image classes, with 1350 images generated per class by SOTA image generators: ADM, BigGAN, glide, Midjourney, stable diffusion v1.4, stable diffusion v1.5, VQDM, and wukong. However, we down-sampled this dataset by randomly selecting 30,000 images for training and 3744 validation and testing samples, ultimately leading to roughly an 80/20/20 split. It is important to note that we used a seed for the random selection of data samples for replicability: a seed of 42 for each split. In addition to our

primary dataset, we generated 10 images from image generation models on which our model was not trained, including Adobe's Firefly, Google's Gemini, and Open AI's DALLE 3 models to test the generalized fingerprint approach [1], which is detailed in our experiments section. Further, we also added 10 real images from the GenImage dataset for each unseen model, specifically ones we didn't use in our original 80/20/20 split, to these testing samples, making our test split have 3804 samples, which still maintains our rough split.

Table 1: Distribution of Train and Validation down-sampled splits

Model	**Train Dataset**		**Validate Dataset**	
	Nature	**AI**	**Nature**	**AI**
ADM	1815 (6.05%)	1935 (6.45%)	237 (6.33%)	231 (6.17%)
BigGAN	1883 (6.28%)	1867 (6.22%)	237 (6.33%)	231 (6.17%)
Midjourney	1866 (6.22%)	1884 (6.28%)	230 (6.14%)	238 (6.36%)
VQDM	1940 (6.47%)	1810 (6.03%)	257 (6.86%)	211 (5.64%)
glide	1859 (6.20%)	1891 (6.30%)	220 (5.88%)	248 (6.62%)
SD V 1.4	1858 (6.19%)	1892 (6.31%)	219 (5.85%)	249 (6.65%)
SD V 1.5	1828 (6.09%)	1922 (6.41%)	234 (6.25%)	234 (6.25%)
wukong	1852 (6.17%)	1898 (6.33%)	215 (5.74%)	253 (6.76%)
Total Samples	*14,901 (49.67%)*	*15,099 (50.33%)*	*1,849 (49.39%)*	*1,895 (50.61%)*

Table 2: Distribution of Test down-sampled and Unseen models split

Model	**Nature**	**AI**
ADM	227 (5.97%)	241 (6.34%)
BigGAN	231 (6.07%)	237 (6.23%)
Midjourney	218 (5.73%)	250 (6.57%)
VQDM	228 (5.99%)	240 (6.31%)
glide	222 (5.84%)	246 (6.47%)
Stable Diffusion V 1.4	231 (6.07%)	237 (6.23%)
Stable Diffusion V 1.5	222 (5.84%)	246 (6.47%)
wukong	227 (5.97%)	241 (6.34%)
DALLE3	10 (0.26%)	10 (0.26%)
Firefly	10 (0.26%)	10 (0.26%)
Gemini	10 (0.26%)	10 (0.26%)
Total Samples	1,879 (49.42%)	1,925(50.58%)

2 Experiments

To ensure that our approach would improve upon the currently established methods, we compared combinations of all the experiments listed below. We tested each combination of the experiment with the following learning rates: 0.001, 0.0001, and 0.00001. Further, we maintained a batch size of 32 for all these experiments.

2.1 Different Pre-Processing Techniques

We tried the following three different pre-processing techniques:

1. Original pre-processing method done by [3].
2. Original Smash and Reconstruct method done by [3] but with a CNN-Learnable Block that accepts 30 high-pass filter feature maps
3. Our new prepossessing technique

2.2 Different Classifiers

This was to test if our SWIN-Transformer v2 would improve upon the existing classifier from [1]. ResNet-50 was added as a baseline.

2.3 Unseen Model vs Seen Models

We introduced images generated by models our model was not trained on to test the generalized fingerprinting technique's effectiveness.

2.4 Unfrozen vs Frozen weights for pre-trained models

The pre-processed images fed to the classifiers differ from traditional images, so we wanted to test if we could get better results by unfreezing the pre-trained models' weights.

3 Results

Table 3: Overall Accuracy and Precision

Experiment	Accuracy	Precision
Swin Transformer v2 Oldest Preprocessing Unfrozen	0.9627	0.9627
Resnet New Preprocessing UnFrozen	0.989	0.9892
Original Paper Model New Preprocessing	0.9876	0.9879
Swin Transformer v2 New Preprocessing UnFrozen	0.9884	0.9886
Resnet Oldest Preprocessing Unfrozen	0.959	0.9591
Swin Transformer v2 Old Smash and Reconstruct UnFrozen	0.9529	0.9538
Resnet Old Smash and Reconstruct UnFrozen	0.9516	0.9523
Swin Transformer v2 New Preprocessing Frozen	0.9658	0.9657
Resnet New Preprocessing Frozen	0.9203	0.9202
Original Paper Model Oldest Preprocessing	0.9876	0.9879
Swin Transformer v2 Oldest Preprocessing Frozen	0.8341	0.8411
Original Paper Model Old Smash and Reconstruct	0.8941	0.897
Swin Transformer v2 Old Smash and Reconstruct Frozen	0.77	0.7704
Resnet Old Smash and Reconstruct Frozen	0.7647	0.7648
Resnet Oldest Preprocessing Frozen	0.7763	0.776

Table 4: Seen Model Accuracy and Precision

Experiment	Accuracy	Precision
Swin Transformer v2 Oldest Preprocessing Unfrozen	0.9688	0.9686
Resnet New Preprocessing UnFrozen	0.9944	0.9944
Original Paper Model New Preprocessing	0.9933	0.9934
Swin Transformer v2 New Preprocessing UnFrozen	0.9947	0.9946
Resnet Oldest Preprocessing Unfrozen	0.9647	0.9647
Swin Transformer v2 Old Smash and Reconstruct UnFrozen	0.9591	0.9595
Resnet Old Smash and Reconstruct UnFrozen	0.9581	0.9584
Swin Transformer v2 New Preprocessing Frozen	0.9736	0.9734
Resnet New Preprocessing Frozen	0.9255	0.9253
Original Paper Model Oldest Preprocessing	0.9933	0.9934
Swin Transformer v2 Oldest Preprocessing Frozen	0.84	0.8465
Original Paper Model Old Smash and Reconstruct	0.9004	0.9028
Swin Transformer v2 Old Smash and Reconstruct Frozen	0.7727	0.7731
Resnet Old Smash and Reconstruct Frozen	0.7687	0.7687
Resnet Oldest Preprocessing Frozen	0.7821	0.7818

Table 5: Unseen Model Accuracy and Precision

Experiment	Accuracy	Precision
Swin Transformer v2 Oldest Preprocessing Unfrozen	0.5833	0.6634
Resnet New Preprocessing UnFrozen	0.65	0.7505
Original Paper Model New Preprocessing	0.6333	0.74
Swin Transformer v2 New Preprocessing UnFrozen	0.6	0.7163
Resnet Oldest Preprocessing Unfrozen	0.6	0.6398
Swin Transformer v2 Old Smash and Reconstruct UnFrozen	0.5667	0.6852
Resnet Old Smash and Reconstruct UnFrozen	0.55	0.6213
Swin Transformer v2 New Preprocessing Frozen	0.4833	0.4722
Resnet New Preprocessing Frozen	0.6	0.6077
Original Paper Model Oldest Preprocessing	0.6333	0.74
Swin Transformer v2 Oldest Preprocessing Frozen	0.4667	0.4534
Original Paper Model Old Smash and Reconstruct	0.5	0.5
Swin Transformer v2 Old Smash and Reconstruct Frozen	0.6	0.6018
Resnet Old Smash and Reconstruct Frozen	0.5167	0.5171
Resnet Oldest Preprocessing Frozen	0.4167	0.4158

4 Conclusion

AI-generated images are becoming increasingly indistinguishable from real images. Our project aims to improve on previous work for detecting AI-generated images. We carry out experiments like using different pre-processing techniques and classifiers, testing on images from seen and unseen models, and with frozen and unfrozen weights. Currently, during pre-processing and feature extraction, we resize the image if one of the dimensions is less than 256px so that we obtain the desired number of patches which leads to loss of texture information. In the future, we hope to find a way to make the process of extracting the inter-pixel correlation contrast more robust to the size of the image. One of the ways we could do this is by implementing an additional classifier for the smaller images to maintain their textural diversity. Further, we use images generated by 8 generative AI models as our training dataset and images generated by an additional 3 for our test dataset. In the future, we plan on training our model on a more diverse, as well as a larger dataset. Our experiments yielded poorer accuracy and precision on the unseen models. So, we hope to expand our test dataset as well, to get more accurate results.

References

[1] Zhong, Nan, et al. "Rich and poor texture contrast: A simple yet effective approach for ai-generated image detection." arXiv preprint arXiv:2311.12397 (2023).
https://arxiv.org/abs/2311.12397

[2] DIRE Repository, accessed April 08, 2024,
https://github.com/ZhendongWang6/DIRE

[3] Hriday Keswani, 2024, "Detection of AI generated images using rich and poor texture contrast", accessed March 19, 2024,
<https://medium.com/@hridaykeswani/detection-of-ai-generated-images-using-rich-and-poor-texture-contrast-fc2024e3e716>

[4] Detection of AI-generated images, accessed April 04, 2024,
https://github.com/hridayK/Detection-of-AI-generated-images

[5] SentryAI,
https://github.com/KostaKat/SentryAI

[6] Zhu, Mingjian, et al. "Genimage: A million-scale benchmark for detecting ai-generated image." Advances in Neural Information Processing Systems 36 (2024).
https://arxiv.org/abs/2306.08571

[7] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, Alexei A. Efros, 2020, "CNN-generated images are surprisingly easy to spot... for now"
https://arxiv.org/abs/1905.11946

[8] Mingxing Tan, Quoc V. Le, 2020, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks"
https://arxiv.org/abs/1912.11035 [9] https://github.com/hridayK/Detection-of-AI-generated-images [10] Swin Transformer v2 by Microsoft
https://huggingface.co/microsoft/swinv2-tiny-patch4-window8-256/blob/main/README.md?code=true