ISITCAP?: Identification of Synthetic Images Through Convolutional Autoencoder Preprocessing

Isha Gokhale
University of Washington
igokhale@uw.edu

Saisriram Gurajala
University of Washington
sgura99@uw.edu

Daniel Vogler
University of Washington
dvogler@uw.edu

1. Abstract

To prevent fraud and misinformation, it is important to be able to tell if images are real or fake. An intrinsic challenge to training robust models is the tendency for synthetic image classifiers to learn small visual imperfections rather than broad visual features. Many studies note significant declines in a model's accuracy when it is tasked with classifying synthetic images created by a different generative model from the ones in the classifier's training set. We attempt to mitigate this phenomenon by integrating autoencoder reconstruction into current image augmentation strategies. To that end, we experiment with several different image conditions: unmodified, perturbed, and reconstructed via autoencoder. We find that in a smaller dataset, convolutional neural networks learn more general features when training images pass through an autoencoder. However, they still experience poor out-of-sample accuracy. Interestingly we find that a more diverse dataset helps classifiers generalize better when shown images unlike the training set across baseline, perturbed, and autoencoder-reconstructed conditions. Taken together, our results suggest a potential for autoencoder reconstruction as a means for augmenting existing datasets. Additionally, autoencoders may promote generalizability to out-of-sample challenge and disperse importance of learned features better than standard augmentation strategies.

2. Introduction

The latest diffusion-based image generation models [6] can output synthetic images that humans struggle to distinguish from real ones [7]. Synthetic images with deceptive potential carry obvious risks of real-world harm; deepfakes and synthetic art that passes for human-created art are just two examples [8]. A recent publication benchmarks human perception for AI generated images, and reports a human misidentification rate of 38.7% [3]. Given the risks accompanying these advances, an active area of computer vision research is training classifiers distinguishing between real and synthetic images for fraud prevention. The most grounded subclass of this problem is distinguishing real images from synthetic look-alikes - synthetic images that appear realistic enough to potentially deceive a human viewer.

To successfully distinguish real images from realistic synthetics, researchers have trained a variety of models [4] on datasets containing real and fake images. In this study, we build on a recent contribution by Bird and Lotfi [1] which introduces a dataset with real and synthetic look-alike images based on CIFAR-10 as well as a convolutional neural network (CNN) classifier that distinguishes them with 92 percent accuracy. We contribute to their body of work by fully characterizing the classifier architecture specified in their dataset. We achieve this thorough characterization through controlled experimentation at baseline (recapitulation of their classifier), with perturbation (like blurring, cropping, and adding a color-jitter), and with an autoencoder image processing step. Additionally, we utilize explainability techniques to characterize feature learning by CNNs for each experiment. Finally, we test model generalizability on unseen datasets and contrast this across different experimental conditions.

3. Related Work

Bird and Lotfi [1] contribute to the literature on classification between real and synthetic images in several ways. Firstly, they use StableDiffusion to generate CIFAKE, a set of synthetic images that closely resemble real images in the extensively-studied CIFAR-10 dataset. They ensure that images are balanced with respect to the 10 classes in CIFAR-10 and their outputs are generally photorealistic. These traits make CIFAKE especially valuable for training models to distinguish real images from synthetic look-alikes. While other projects juxtapose real and fake images, this dataset balanced real and fake images with respect to subject matter. Bird and Lotfi train a convolutional neural network to classify images in this data into real vs.

synthetic with 92 percent accuracy [1]. Finally, the authors employ explainable AI (xAI) techniques to visualize how the model decides whether an image is real or synthetic. [1].

While Bird and Lotfi make an important contribution with CIFAKE and their analysis of how CNNs distinguish between real and fake CIFAKE images, we see room for methodological improvements to test whether their findings are generalizable. Stanciu et. al point out that classifiers like Bird and Lotfi's often work well on the dataset on which they are trained, which have images from a specific synthetic generation model, but "do not yield good results when evaluating on samples coming from other distributions" [7]. Beyond deepfake detection, Lu et. al challenged general fake image detectors with diffusion model versions outside of the training distribution and observed a significant drop off in performance [3]. We interpret this as evidence that classifiers learn signatures of the specific model used to produce the synthetic images rather than learning a signature for synthetic images in general.

One way to mitigate the influence of small visual imperfections on classifier output is to add noise through random perturbations to images. But Bird and Lotfi do not report applying any common data perturbation techniques, such as random blurs, crops, or perturbations to the real section of their dataset. Saliency maps generated on Bird and Lotfi's CNN model show the classifier's dependence on small visual imperfections or "glitches" for fake image determination. As generative models improve, it is likely that these glitches will be mitigated, so future real/synthetic classifiers will not be able to rely on them. Therefore, we found this to be a significant methodological limitation; if real images were randomly perturbed, glitches similar to the ones found in the synthetic images would probability also appear in the real images (or may be obscured in the fake images), and the model would no longer rely on them as heavily. Instead, to be successful, the model would have to learn more inherent distinctions between real and synthetic images — features that occur throughout the image — instead of relying on visual glitches. We hypothesized that adding perturbation steps to pre-processing would improve the generalizability of the baseline classifier, although we expected it to decrease in-sample performance.

Stanciu and Ionescu agree that "augment[ing] the training dataset [helps] to increase generalization," and they apply a further augmentation method: "using autoencoders to replicate the image with a slight error… add[s] some noise of frequency components to the image while maintaining its quality" [7]. Ojha et. al [5] experiment with this, and find that the generalizability gap is caused by classifiers learning these same spectral artifacts from diffusion generation. To that end, they train upon a feature space not attached to a specific classification objective, such as ViT-L. We attempt to merge these objectives via convolutional autoencoder preprocessing: a basal reconstruction of the input images separate from a real or fake classification. In an attempt to improve the generalizability of classifiers like Bird and Lotfi's, we replicate their basic classifier architecture but also apply random crops, blurs, pixel perturbations and reconstruction via autoencoder to CIFAKE. We expect that these changes will force the classifier to learn more generalizable features for real/synthetic detection. We employed similar explainable AI techniques to assess whether these sorts of perturbations reduce model dependence on glitches. Finally, in a second phase of experimentation we integrate CIFAKE into a more diverse training dataset also including images from Fake2M [3].

Table 1. Phase 1

Data Sets	Real Images	Fake Images
Train and Val Sets		
CIFAKE	55,000	55,000
Test Sets in-sample		
CIFAKE	5,000	5,000
Test Sets out-of-sample		
ImageNet	1,000	-
SDv1.5	-	798
ImageNet	1,000	-
IFv1	-	1,000

4. Methods

4.1. Data Collection and Preparation

We downloaded CIFAKE from a Kaggle repository hosted by the original authors using the Python API[1]. We scraped tiny-imagenet through similar means[2]. We also sourced images generated by the SDv1.5, SDv2.1, IFv1, and CogView2 datasets made available through the SentryImage project,[3] associated with [3] through HuggingFace. All CIFAKE images were 32 x 32 pixels, the same dimensions as CIFAR-10. For images from the other diffusion models that were part of our out-of-sample test sets, which did not originate at 32x32 height by width, we applied resizing. We scaled pixel values from the standard range of zero to 255 to zero to one. Prior to input into the autoencoder, we scaled pixel values to lie between -1 and

[1] https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images

[2] https://www.kaggle.com/competitions/tiny-imagenet

[3] https://github.com/Inf-imagine/Sentry

Table 2. Phase 2

Data Sets	Real Images	Fake Images
Train and Val Sets		
CIFAKE	27,500	27,500
SDv1.5	-	22.000
IFv1	-	5,500
ImageNet	27,500	-
Test Sets in-sample		
CIFAKE	2,500	2,500
ImageNet	2,500	-
SDv1.5	-	500
IFv1	2,000	-
Test Sets out-of-sample		
SDv2.1	-	1,000
ImageNet	1,000	-
Cogview2	-	1,000
ImageNet	1,000	-

1. Tables 1 and 2 summarize the training and test sets we used in the two phases of experimentation.

4.2. Baseline CNN Replication

Using PyTorch, we recapitulated Bird and Lotfi's classifier model architecture. We implemented two convolutional, non-linear ReLU, and max pool sandwich layers. In each convolutional layer we utilized 32 3x3 kernels with padding 1 and stride 1 to preserve input dimensions. Each max pool operation was done with a 3x3 kernel with stride 1 and padding 1. After the last max pool operation, we implemented three dense layers of size 64 x 64, 64 x 64, and 64 x 1, respectively. A sigmoid function was used at the end of the network, and loss was defined using binary cross entropy. We performed a hyperparameter sweep and always report the results of the classifier CNN with the highest validation accuracy after tuning.

4.3. Data Processing: Perturbation

We applied Gaussian blur, random resized crop, and color jitter transformations as standard perturbations. The probability of applying all of these transformation on the images was 80%.

4.4. Data Processing: Autoencoder Architecture

We implemented an autoencoder through PyTorch.[4] Our best-performing architecture included an encoder

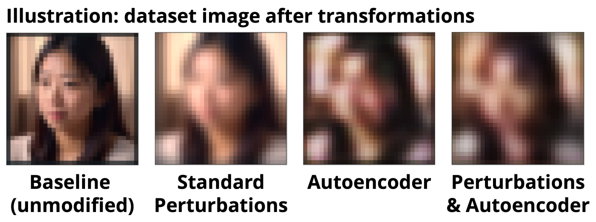
Illustration: dataset image after transformations
Baseline (unmodified)
Standard Perturbations
Autoencoder
Perturbations & Autoencoder

Figure 1. Illustration of a randomly sampled Phase 1 training set image under four experimental conditions

constructed as follows: 32 3x3 convolutional kernels with stride two and padding one, a ReLU nonlinearity, 64 3x3 convolutional kernels with stride two and padding one, and a ReLU non-linearity. Inputs were then flattened, and a dense layer converted outputs from size 4096 to a latent representation of size 2048. The decoder first uses a dense linear layer to convert the latent representation back to size 4096. Then bilinear upsampling is performed with a scale factor of 2. Up-sampling is followed by a transposed convolution with 32 3x3 kernels of size with stride 1 and padding 1, and a ReLU non-linearity is applied. We again apply bilinear upsampling with scale factor 2, transposed convolution with 3 3x3 kernels with same stride and padding as before, and end with a `tanh` nonlinearity. We measured loss through mean squared error of the reconstructed image relative to the input.

4.5. Quantifying Accuracy

Throughout this study, we quantify the classifier's accuracy under different conditions by running inference and computing the percentage of correct predictions using two different test sets: in-sample and out-of-sample. **In-sample test set** refers to a set of images created by the **same generative model** as the one that created the images in the classifier's training set. **Out-of-sample test set** refers to a set of images created by a **different generative model** from the one that created the images used in training. We say that a classifier generalizes well when its out-of-sample performance is approximately equal to its in-sample performance.

4.6. Explainability through Saliency Maps

We created saliency maps for each variant of the CNN classifier that we trained. We used one image from each of our test sets (CIFAKE, and our two out-of-sample datasets) as our input images for the saliency map. We created saliency maps by computing the gradients of the image from each of our test sets with respect to the maximum score of the image.

[4]Our autoencoder architecture was inspired by https://github.com/rtflynn/Cifar-Autoencoder, but implemented independently

5. Experiments

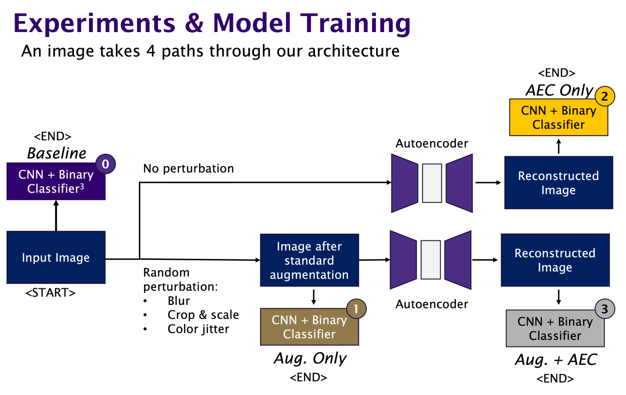

Figure 2. Images pass through our classifier through four conditions: unmodified (baseline), perturbed, autoencoder reconstruction, and both perturbed and autoencoder reconstruction

We conducted experiments in two phases. In the first phase, we trained a CNN with the same architecture as [1] only on CIFAKE data and assessed its accuracy under various experimental conditions in and out-of-sample. In the second phase, we re-trained the model with a more diverse training set (see Data Collection and Preparation) and again assessed accuracy in and out-of-sample.

5.1. Phase 1

Our experimental design (**Figure 2**) involved training on CIFAKE images across four experimental conditions: (1) Baseline — no perturbations or autoencoder reconstruction, (2) perturbations only, (3) autoencoder reconstruction only, (4) perturbation and autoencoder reconstruction (**Figure 1**). We then measured test split accuracy across our four conditions. We expected lower performance in these conditions as the images were being perturbed or reconstructed, obscuring ordinarily facile learning of visual imperfections.

We next tested the model on two different test sets coming from out-of-sample distributions: Stable Diffusion v15 and DeepFloyd IF v1 (**Table 1**). We expected the perturbation, and especially the autoencoder reconstruction, to help the classifier generalize well. We hypothesized that, for any given condition, its out-of-sample performance would approximately equal its in-sample accuracy. Following the test run, we compared the classifier's accuracy between in-sample (CIFAKE) and out-of-sample (StableDiffusion v1.5 and IF v1) across the four experimental conditions. We additionally generated saliency maps, enabling us to analyze whether the classifier learns general or specific features.

5.2. Phase 2

We re-trained our baseline classifier using a more diverse dataset (**Table 2**), and replicated the same experimental conditions from Phase 1. However, we included the datasets generated by CIFAKE, StableDiffusion v1.5 and DeepFloyd IF v1 images in the training set and considered them to be in training sample distribution. We defined two datasets, generated by StableDiffusion v2.1 and CogView2, to be out-of-sample. Once again, we compared the classifier's accuracy between in-sample (CIFAKE/SDv1.5/IF) and out-of-sample (other generative models) across the four experimental conditions and generated saliency maps.

6. Results

Table 3. Phase 1 Model Accuracy on Test Sets

Condition	CIFAKE	SDv1.5	IFv1
Baseline	93.8%	73.9%	63.4%
Perturbations	87.0%	62.2%	54.3%
AEC	87.6%	66.0%	54.9%
Aug. + AEC	85.4%	54.4%	51.3%

In training datasets from all generative models, we observe a significant drop in accuracy when the classifier is tested on our two out-of-sample datasets (each generated by two different diffusion models). The baseline model outperforms all other models on all the test sets. We note that the autoencoder-only model performs 11.6% better on the SDv1.5 out-of-sample test set than the perturbation and autoencoder reconstruction models. We additionally find performance to be higher (+3.8%) for the autoencoder condition relative to the perturbation condition for out-of-sample test set SDv1.5. We find that condition (3), image perturbation and autoencoder reconstruction, results in the lowest performance (**Table 3**). Examination of saliency maps across conditions suggests differences in dispersion of feature importance across different conditions (**Figure 3**). We note a recapitulation of Bird and Lotfi's results through capture of specific visual imperfections in the third row. Qualitatively, we find that the addition of autoencoders disperses the feature importance across the saliency map the most.

In Phase 2, we saw a similar absolute performance drop when comparing baseline model performance in-sample relative to performance after perturbation or autoencoder reconstruction. However, we note more generalizable

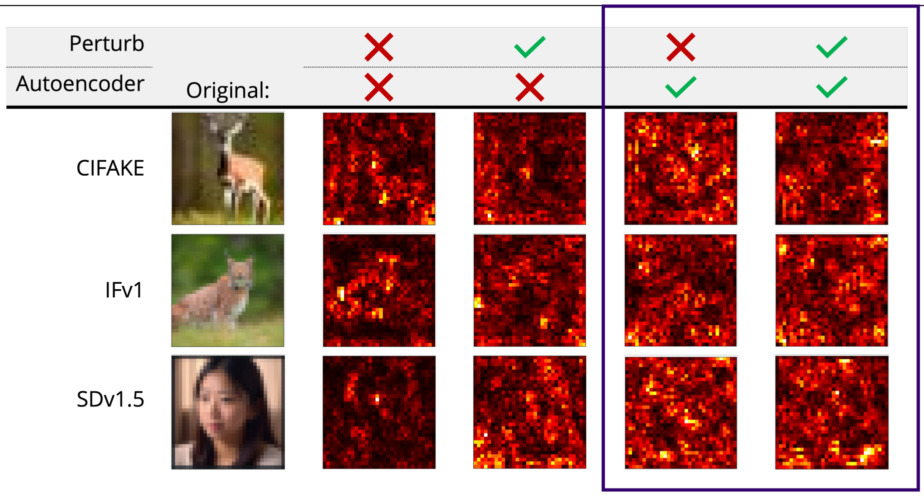

Figure 3. Illustration of which image regions the classifier uses to decide whether an image is real or fake from datasets generated by three different models. Passing images through autoencoder seems to cause classifier to learn more background features.

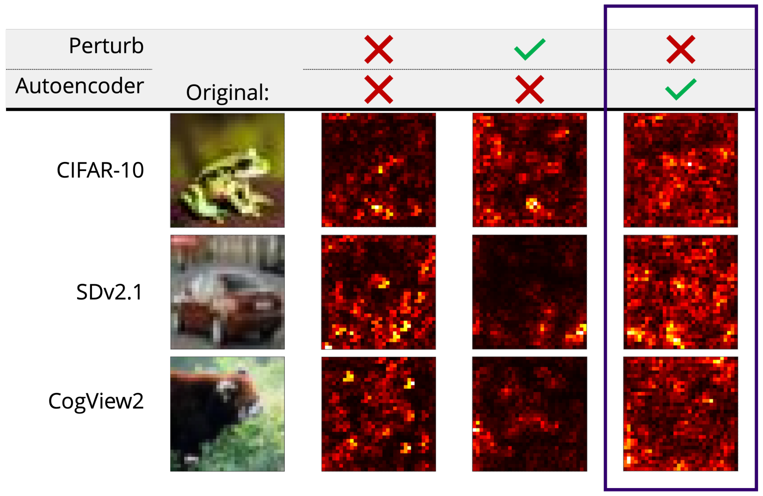

Figure 4. Illustration of which image regions the classifier uses to uses to decide whether an image is real or fake from datasets generated by three different models. CIFAR-10 image is a real image.

Table 4. Phase 2 Model Accuracy on Test Sets

Condition	in-sample	SDv2.1	Cogview2
Baseline	91.3%	93.9%	86.6%
Perturbations	77.2%	76.7%	65.4%
AEC	79.8%	76.7%	**71.4%**

performance on out-of-sample test-sets. Specifically, we find SDv2.1 to preserve performance well across all experimental conditions. Cogview2, however, precipitates a steep decline in performance for perturbation, (-11.8%). We see a 6% increase in performance on Cogview2 when comparing autoencoder reconstructed image classification relative to standard perturbation. Baseline remains the highest-performing model **(Table 4)**. Using saliency mapping with Phase 2 models displays similar findings to Phase 1: clear pockets of feature importance at baseline. However, we find that standard perturbation in this case condenses the feature importance more than in Phase 1 for out-of-sample. Saliency mapping again suggests that autoencoder reconstruction disperses feature importance (**Figure 4**).

7. Discussion

In phase one of our study, in which the classifier was trained only on CIFAKE-10 real and fake images, we noted significant performance decline when testing the model on out-of-sample images relative to performance in-sample. Although saliency maps do suggest that the transformations we applied caused the classifier to learn more general features (Figure 3), this did not translate into strong prediction accuracy out-of-sample.

We think of this as a function of two dataset characteristics: dataset diversity and dataset depth. More specific applications of generative image detection contend with narrow dataset diversity (e.g., deepfake detection needs only to learn features unique to faces). However, abstracting generative image detection across many classes requires learning a breadth of features characterizing diverse image classes. The CIFAKE dataset, while high quality and photo-realistic, was not sufficiently diverse and deep. Additionally, the learned features likely did not translate well to non CIFAR-10 image classes. Put simply, the significant drop in performance for out-of-sample tests sets led us to conclude that applying perturbations and autoencoders does not adequately compensate insufficient dataset diversity.

To combat this, in phase two we test classifiers trained on datasets with a broader range of image classes and produced by various generative models. Accordingly, in phase two we found that classifiers generalize better to out-of-sample test sets when trained on a more diverse dataset. Specifically, we note that the out-of-sample SDv2.1 results in comparable performance to in-sample test sets. This finding may be explained by a certain degree of similarity between the in-sample training set (SDv1.5) and the out-of-sample set (SDv2.1). However, based on the performance uplift relative to standard perturbation we conclude that autoencoder reconstruction may represent a potential avenue for improving model generalizability. The relatively high performance on the CogView and SDv2.1 test set can be attributed to sourcing the real images from TinyImageNet, which the model was trained on. However, these real images were not included in the in-sample training, validation, and test sets. We therefore attribute across the board higher performance on out-of-sample test sets to an increase in dataset diversity for each model.

Extension of this study requires in-depth characterization of autoencoder reconstruction and convolutional feature extraction across a deep and diverse dataset. We conclude that autoencoders are potential avenues for improving generalizability of synthetic image classifiers. Moving forward, we would like to conduct experiments better measuring the efficacy of autoencoders for data augmentation. Future steps may include use of pretrained autoencoders for image reconstruction and pretrained convolutional nets for feature extraction. We find these changes may present new avenues for augmentation as autoencoder reconstruction represents a differentiable, trainable process for generating altered images. Usage of pretrained CNNs such as VGGnet [9] and ResNet [2] may improve generalizability as features would be extracted independent of the rate-limiting real/fake objective noted by Ojha et. al [5].

References

[1] Jordan J. Bird and Ahmad Lotfi. Cifake: Image classification and explainable identification of ai-generated synthetic images, 2023. 1, 2, 4

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 6

[3] Zeyu Lu, Di Huang, LEI BAI, Jingjing Qu, Chengyue Wu, Xihui Liu, and Wanli Ouyang. Seeing is not always believing: Benchmarking human and model perception of ai-generated images. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 25435–25447. Curran Associates, Inc., 2023. 1, 2

[4] Xuan Ni, Linqiang Chen, Lifeng Yuan, Guohua Wu, and Ye Yao. An evaluation of deep learning-based computer generated image detection approaches. *IEEE Access*, 7:130830–130840, 2019. 1

[5] Utkarsh Ojha, Yuheng Li, and Yong Jae Lee. Towards universal fake image detectors that generalize across generative models. In *CVPR*, 2023. 2, 6

[6] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022. 1

[7] Dan-Cristian Stanciu and Bogdan Ionescu. Autoencoder-based data augmentation for deepfake detection. In *Proceedings of the 2nd ACM International Workshop on Multimedia AI against Disinformation*, ICMR '23. ACM, June 2023. 1, 2

[8] Mulin Tian, Mahyar Khayatkhoei, Joe Mathai, and Wael AbdAlmageed. Unsupervised multimodal deepfake detection using intra- and cross-modal inconsistencies, 2023. 1

[9] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017. 6