Towards Universal Fake Image Detectors that Generalize Across Generative Models

Utkarsh Ojha* Yuheng Li* Yong Jae Lee

University of Wisconsin-Madison

Abstract

With generative models proliferating at a rapid rate, there is a growing need for general purpose fake image detectors. In this work, we first show that the existing paradigm, which consists of training a deep network for real-vs-fake classification, fails to detect fake images from newer breeds of generative models when trained to detect GAN fake images. Upon analysis, we find that the resulting classifier is asymmetrically tuned to detect patterns that make an image fake. The real class becomes a 'sink' class holding anything that is not fake, including generated images from models not accessible during training. Building upon this discovery, we propose to perform real-vs-fake classification without learning*; i.e., using a feature space not explicitly trained to distinguish real from fake images. We use nearest neighbor and linear probing as instantiations of this idea. When given access to the feature space of a large pretrained vision-language model, the very simple baseline of nearest neighbor classification has surprisingly good generalization ability in detecting fake images from a wide variety of generative models; e.g., it improves upon the SoTA [50] by* **+15.07 mAP** *and* **+25.90% acc** *when tested on unseen diffusion and autoregressive models. Our code, models, and data can be found at* https://github.com/Yuheng-Li/UniversalFakeDetect

1. Introduction

The digital world finds itself being flooded with many kinds of fake images these days. Some could be natural images that are doctored using tools like Adobe Photoshop [1, 49], while others could have been generated through a machine learning algorithm. With the rise and maturity of deep generative models [22,29,42], fake images of the latter kind have caught our attention. They have raised excitement because of the quality of images one can generate with ease. They have, however, also raised concerns about their use for malicious purposes [4]. To make matters worse, there is no longer a single source of fake images that needs to be dealt with: for example, synthesized images could take the form of realistic human faces generated using generative adversarial networks [29], or they could take the form of complex scenes generated using diffusion models [42, 45]. One can be almost certain that there will be more modes of fake images coming in the future. With such a diversity, our goal in this work is to develop a general purpose fake detection method which can detect whether any arbitrary image is fake, given access to only one kind of generative model during training; see Fig. 1.

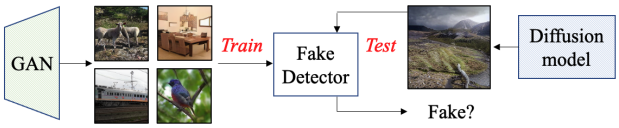

Figure 1. Using images from just one generative model, can we detect images from a different type of generative model as fake?

A common paradigm has been to frame fake image detection as a learning based problem [10, 50], in which a training set of fake and real images are assumed to be available. A deep network is then trained to perform real vs fake binary classification. During test time, the model is used to detect whether a test image is real or fake. Impressively, this strategy results in an excellent generalization ability of the model to detect fake images from different algorithms *within the same generative model family* [50]; e.g., a classifier trained using real/fake images from ProGAN [28] can accurately detect fake images from StyleGAN [29] (both being GAN variants). However, to the best of our knowledge, prior work has not thoroughly explored generalizability across different families of generative models, especially to ones not seen during training; e.g., will the GAN fake classifier be able to detect fake images from diffusion models as well? Our analysis in this work shows that existing methods *do not* attain that level of generalization ability.

Specifically, we find that these models work (or fail to work) in a rather interesting manner. Whenever an image contains the (low-level) fingerprints [25, 50, 52, 53] particu-

*Equal contribution

lar to the generative model used for training (e.g., ProGAN), the image gets classified as fake. *Anything else* gets classified as real. There are two implications: (i) even if diffusion models have a fingerprint of their own, as long as it is not very similar to GAN's fingerprint, their fake images get classified as real; (ii) the classifier doesn't seem to look for features of the real distribution when classifying an image as real; instead, the real class becomes a 'sink class' which hosts anything that is not GAN's version of fake image. In other words, the decision boundary for such a classifier will be closely bound to the particular fake domain.

We argue that the reason that the classifier's decision boundary is unevenly bound to the fake image class is because it is easy for the classifier to latch onto the low-level image artifacts that differentiate fake images from real images. Intuitively, it would be easier to learn to spot the fake pattern, rather than to learn all the ways in which an image could be real. To rectify this undesirable behavior, we propose to perform real-vs-fake image classification using features that are *not trained* to separate fake from real images. As an instantiation of this idea, we perform classification using the *fixed* feature space of a CLIP-ViT [24, 41] model pre-trained on internet-scale image-text pairs. We explore both nearest neighbor classification as well as linear probing on those features.

We empirically show that our approach can achieve significantly better generalization ability in detecting fake images. For example, when training on real/fake images associated with ProGAN [28] and evaluating on unseen diffusion and autoregressive model (LDM+Glide+Guided+DALL-E) images, we obtain improvements over the SoTA [50] by (i) **+15.05mAP and +25.90% acc** with nearest neighbor and (ii) **+19.49mAP and +23.39% acc** with linear probing. We also study the ingredients that make a feature space effective for fake image detection. For example, can we use any image encoder's feature space? Does it matter what domain of fake/real images we have access to? Our key takeaways are that while our approach is robust to the breed of generative model one uses to create the feature bank (e.g., GAN data can be used to detect diffusion models' images and vice versa), one needs the image encoder to be trained on internet-scale data (e.g., ImageNet [21] does not work).

In sum, our main contributions are: (1) We analyze the limitations of existing deep learning based methods in detecting fake images from unseen breeds of generative models. (2) After empirically demonstrating prior methods' ineffectiveness, we present our theory of what could be wrong with the existing paradigm. (3) We use that analysis to present two very simple baselines for real/fake image detection: nearest neighbor and linear classification. Our approach results in state-of-the-art generalization performance, which even the oracle version of the baseline (tuning its confidence threshold on the *test set*) fails to reach. (4) We thoroughly study the key ingredients of our method which are needed for good generalizability.

2. Related work

Types of synthetic images. One category involves altering a portion of a real image, and contains methods which can change a person's attribute in a source image (e.g., smile) using Adobe's photoshop tool [1, 39], or methods which can create DeepFakes replacing the original face in a source image/video with a target face [2, 3]. Another recent technique which can optionally alter a part of a real image is DALL-E 2 [42], which can insert an object (e.g., a chair) in an existing real scene (e.g., office). The other category deals with any algorithm which generates all pixels of an image from scratch. The input for generating such images could be random noise [28, 29], categorical class information [7], text prompts [31, 36, 42, 46], or could even by a collection of images [32]. In this work, we consider primarily this latter category of generated images and see if different detection methods can classify them as fake.

Detecting synthetic images. The need for detecting fake images has existed even before we had powerful image generators. When traditional methods are used to manipulate an image, the alteration in the underlying image statistics can be detected using hand-crafted cues such as compression artifacts [5], resampling [40] or irregular reflections [37]. Several works have also studied GAN synthesized images in their frequency space and have demonstrated the existence of much clearer artifacts [25, 53].

Learning based methods have been used to detect manipulated images as well [15, 44, 49]. Earlier methods studied whether one can even learn a classifier that can detect other images from the same generative model [25, 34, 47], and later work found that such classifiers do not generalize to detecting fakes from other models [19, 53]. Hence, the idea of learning classifiers that generalize to other generative models started gaining attention [17, 35]. In that line of work, [50] proposes a surprisingly simple and effective solution: the authors train a neural network on real/fake images from one kind of GAN, and show that it can detect images from other GAN models as well, if an appropriate training data source and data augmentations are used. [10] extends this idea to detect patches (as opposed to whole images) as real/fake. [6] investigates a related, but different, task of predicting which of two test images is real and which one is modified (fake). Our work analyses the paradigm of training neural networks for fake image detection, showing that their generalizability does not extend to unseen families of generative models. Drawing on this finding, we show the effectiveness of a feature space *not explicitly learned* for the task of fake image detection.

3. Preliminaries

Given a test image, the task is to classify whether it was captured naturally using a camera (real image) or whether it was synthesized by a generative model (fake image). We first discuss the existing paradigm for this task [10, 50], the analysis of which leads to our proposed solution.

3.1. Problem setup

The authors in [50] train a convolutional network (f) for the task of binary real (0) vs fake (1) classification using images associated with one generative model. They train ProGAN [28] on 20 different object categories of LSUN [51], and generate 18k fake images per category. In total, the real-vs-fake training dataset consists of 720k images (360k in *real* class, 360k in *fake* class). They choose ResNet-50 [27] pretrained on ImageNet [21] as the fake classification network, and replace the fully connected layer to train the network for real vs fake classification with the binary cross entropy loss. During training, an intricate data augmentation scheme involving Gaussian blur and JPEG compression is used, which is empirically shown to be critical for generalization. Once trained, the network is used to evaluate the real and fake images from other generative models. For example, BigGAN [7] is evaluated by testing whether its class-conditioned generated images (F_{BigGAN}) and corresponding real images (R_{BigGAN}: coming from ImageNet [21]) get classified correctly; i.e., whether $f(R_{BigGAN}) \approx 0$ and $f(F_{BigGAN}) \approx 1$. Similarly, each generative model (discussed in more detail in Sec. 5.1) has a test set with an equal number of real and fake images associated with it.

3.2. Analysis of why prior work fails to generalize

We start by studying the ability of this network—which is trained to distinguish ProGAN fakes from real images—to detect generated images from unseen methods. In Table 1, we report the accuracy of classifying the real and fake images associated with different families of generative models. As was pointed out in [50], when the target model belongs to the same breed of generative model used for training the real-vs-fake classifier (i.e., GANs), the network shows good overall generalizability in classifying the images; e.g., GauGAN's real/fake images can be detected with 79.25% accuracy. However, when tested on a different family of generative models, e.g., LDM and Guided (variants of diffusion models; see Sec. 5.1), the classification accuracy drastically drops to near *chance* performance![1]

Now, there are two ways in which a classifier can achieve chance performance when the test set has an equal number of real and fake images: it can output (i) a random prediction for each test image, (ii) the same class prediction for all test images. From Table 1, we find that for diffusion models, the classifier works in the latter way, classifying *almost all* images as real regardless of whether they are real (from LAION dataset [48]) or generated. Given this, it seems f has learned an *asymmetric* separation of real and fake classes, where for any image from either LDM (unseen fake) or LAION (unseen real), it has a tendency to disproportionately output one class (real) over the other (fake).

[1]Corresponding precision-recall curves can be found in the appendix.

	CycleGAN	GauGAN	LDM	Guided	DALL-E
Real acc.	98.64	99.4	99.61	99.14	99.61
Fake acc.	62.91	59.1	3.05	4.67	4.9
Average	80.77	79.25	51.33	51.9	52.26
Chance performance	50.00	50.00	50.00	50.00	50.00

Table 1. Accuracy of a real-vs-fake classifier [50] trained on ProGAN images in detecting real and fake images from different types of generative models. LDM, Guided, and DALL-E represent the breeds of image generation algorithms not seen during training.[1]

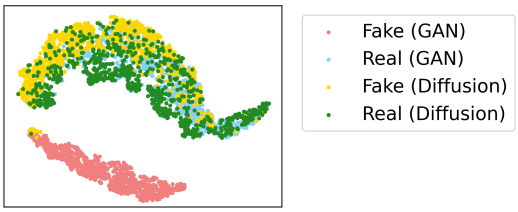

Figure 2. t-SNE visualization of real and fake images associated with two types of generative models. The feature space used is of a classifier trained to distinguish Fake (GAN) from Real (GAN).

To further study this unusual phenomenon, we visualize the feature space used by f for classification. We consider four image distributions: (i) F_{GAN} consisting of fake images generated by ProGAN, (ii) R_{GAN} consisting of the real images used to train ProGAN, (iii) $F_{Diffusion}$ consisting of fake images generated by a latent diffusion model [46], and (iv) $R_{Diffusion}$ consisting of real images (LAION dataset [48]) used to train the latent diffusion model. The real-vs-fake classifier is trained on (i) and (ii). For each, we obtain their corresponding feature representations using the penultimate layer of f, and plot them using t-SNE [33] in Fig. 2. The first thing we notice is that f indeed does not treat real and fake classes equally. In the learned feature space of f, the four image distributions organize themselves into two noticeable clusters. The first cluster is of F_{GAN} (pink) and the other is an amalgamation of the remaining three (R_{GAN} + $F_{Diffusion}$ + $R_{Diffusion}$). In other words, f can easily distinguish F_{GAN} from the other three, but the learned real class does not seem to have any property (a space) of its own, but is rather used by f to form a *sink class*, which hosts anything that is not F_{GAN}. The second thing we notice is that the cluster surrounding the learned fake class is very condensed compared to the one surrounding the learned real class, which is much more open. This indicates that f can detect a common property among im-

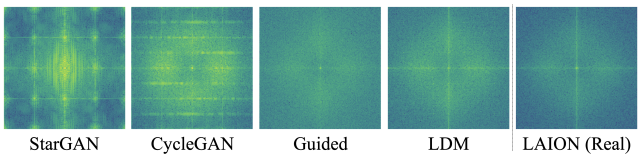

Figure 3. Average frequency spectra of each domain. The first four correspond to fake images from GANs and diffusion models. The last one represents real images from LAION [48] dataset.

ages from F_{GAN} with more ease than detecting a common property among images from R_{GAN}.

But why is it that the property that f finds to be common among F_{GAN} is useful for detecting fake images from other GAN models (e.g., CycleGAN), but not for detecting $F_{Diffusion}$? In what way are fake images from diffusion models different than images from GANs? We investigate this by visualizing the frequency spectra of different image distributions, inspired by [8, 9, 50, 53]. For each distribution (e.g., F_{BigGAN}), we start by performing a high pass filtering for each image by subtracting from it its median blurred image. We then take the average of the resulting high frequency component across 2000 images, and compute the Fourier transform. Fig. 3 shows this average frequency spectra for four fake domains and one real domain. Similar to [50], we see a distinct and repeated pattern in StarGAN and CycleGAN. However, this pattern is missing in the fake images from diffusion models (Guided [23] and LDM [46]), similar to images from a real distribution (LAION [48]). So, while fake images from diffusion models seem to have some common property of their own, Fig. 3 indicates that that property is not of a similar nature as the ones shared by GANs.

Our hypothesis is that when f is learning to distinguish between F_{GAN} and R_{GAN}, it latches onto the artifacts depicted in Fig. 3, learning only to look for the presence/absence of those patterns in an image. Since this is sufficient for it to reduce the training error, it largely ignores learning any features (e.g., smooth edges) pertaining to the *real* class. This, in turn, results in a skewed decision boundary where a fake image from a diffusion model, lacking the GAN's fingerprints, ends up being classified as real.

4. Approach

If learning a neural network f is not an ideal way to separate real ($\mathcal{R}$) and fake ($\mathcal{F}$) classes, what should we do? The key, we believe, is that the classification process should happen in a feature space which has *not been learned* to separate images from the two classes. This might ensure that the features are not biased to recognize patterns from one class disproportionately better than the other.

Choice of feature space. As an initial idea, since we might not want to learn any features, can we simply perform the classification in pixel space? This would not work, as pixel space would not capture any meaningful information (e.g., edges) beyond point-to-point pixel correspondences. So, any classification decision of an image should be made after it has been mapped into some feature space. This feature space, produced by a network and denoted as ϕ, should have some desirable qualities.

First, ϕ should have been exposed to a large number of images. Since we hope to design a general purpose fake image detector, its functioning should be consistent for a wide variety of real/fake images (e.g., a human face, an outdoor scene). This calls for the feature space of ϕ to be heavily populated with different kinds of images, so that for any new test image, it knows how to embed it properly. Second, it would be beneficial if ϕ, while being general overall, can also capture low-level details of an image. This is because differences between real and fake images arise particularly at low-level details [10, 53].

To satisfy these requirements, we consider leveraging a large network trained on huge amounts of data, as a possible candidate to produce ϕ. In particular, we choose a variant of the vision transformer, ViT-L/14 [24], trained for the task of image-language alignment, CLIP [41]. CLIP:ViT is trained on an extraordinarily large dataset of 400M image-text pairs, so it satisfies the first requirement of sufficient exposure to the visual world. Additionally, since ViT-L/14 has a smaller starting patch size of 14 $\times$ 14 (compared to other ViT variants), we believe it can also aid in modeling the low-level image details needed for real-vs-fake classification. Hence, for all of our main experiments, we use the last layer of CLIP:ViT-L/14's visual encoder as ϕ.

The overall approach can be formalized in the following way. We assume access to images associated with a single generative model (e.g., ProGAN, which is the same constraint as in [50]). $\mathcal{R} = \{r_1, r_2, ..., r_N\}$, and $\mathcal{F} = \{f_1, f_2, ..., f_N\}$ denote the real and fake classes respectively, each containing N images. $\mathcal{D} = \{\mathcal{R} \cup \mathcal{F}\}$ denotes the overall training set. We investigate two simple classification methods: nearest neighbor and linear probing. Importantly, both methods utilize a feature space that is entirely untrained for real/fake classification.

Nearest neighbor. Given the pre-trained CLIP:ViT visual encoder, we use its final layer ϕ to map the entire training data to their feature representations (of 768 dimensions). The resulting feature bank is $\phi_{bank} = \{\phi_{\mathcal{R}} \cup \phi_{\mathcal{F}}\}$ where $\phi_{\mathcal{R}} = \{\phi_{r_1}, \phi_{r_2}, ..., \phi_{r_N}\}$ and $\phi_{\mathcal{F}} = \{\phi_{f_1}, \phi_{f_2}, ..., \phi_{f_N}\}$. During test time, an image x is first mapped to its feature representation ϕ_x. Using cosine distance as the metric d, we find its nearest neighbor to both the real ($\phi_{\mathcal{R}}$) and fake ($\phi_{\mathcal{F}}$) feature banks. The prediction—real:0, fake:1—is given based

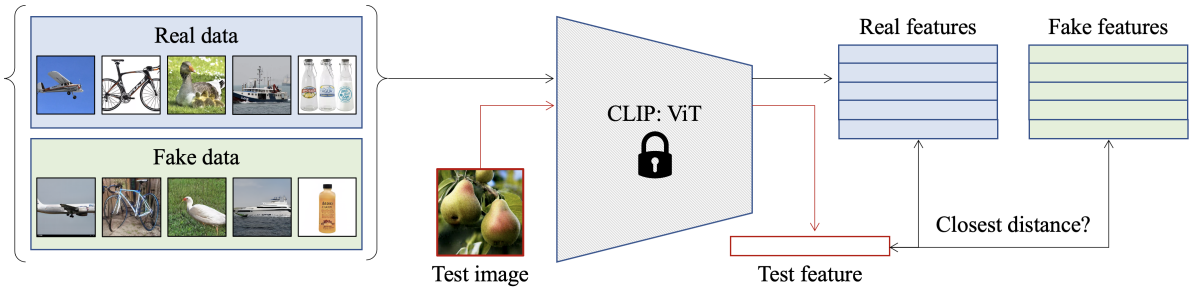

Figure 4. **Nearest neighbors for real-vs-fake classification.** We first map the real and fake images to their corresponding feature representations using a pre-trained CLIP:ViT network *not trained for this task*. A test image is mapped into the same feature space, and cosine distance is used to find the closest member in the feature bank. The label of that member is the predicted class.

on the smaller distance of the two:

$$\text{pred}(x) = \begin{cases} 1, & \text{if } \min_i \left(d(\phi_x, \phi_{f_i})\right) < \min_i \left(d(\phi_x, \phi_{r_i})\right) \\ 0, & \text{otherwise.} \end{cases}$$

The CLIP:ViT encoder is always kept frozen; see Fig. 4.

Linear classification. We take the pre-trained CLIP:ViT encoder, and add a single linear layer with sigmoid activation on top of it, and train *only* this new classification layer ψ for binary real-vs-fake classification using binary cross entropy loss:

$$\mathcal{L} = -\sum_{f_i \in \mathcal{F}} \log(\psi(\phi_{f_i})) - \sum_{r_i \in \mathcal{R}} \log(1 - \psi(\phi_{r_i})).$$

Since such a classifier involves training only a few hundred parameters in the linear layer (e.g., 768), conceptually, it will be quite similar to nearest neighbor and retain many of its useful properties. Additionally, it has the benefit of being more computation and memory friendly.

5. Experiments

We now discuss the experimental setup for evaluating the proposed method for the task of fake image detection.

5.1. Generative models studied

Since new methods of creating fake images are always coming up, the standard practice is to limit access to only one generative model during training, and test the resulting model on images from unseen generative models. We follow the same protocol as described in [50] and use ProGAN's real/fake images as the training dataset.

During evaluation, we consider a variety of generative models. First, we evaluate on the models used in [50]: ProGAN [28], StyleGAN [29], BigGAN [7], CycleGAN [54], StarGAN [13], GauGAN [38], CRN [12], IMLE [30], SAN [18], SITD [11], and DeepFakes [47]. Each generative model has a collection of real and fake images. Additionally, we evaluate on guided diffusion model [23], which is trained for the task for class conditional image synthesis on the ImageNet dataset [21]. We also perform evaluation on recent text-to-image generation models: (i) Latent diffusion model (LDM) [46] and (ii) Glide [36] are variants of diffusion models, and (iii) DALL-E [43] is an autoregressive model (we consider its open sourced implementation DALL-E-mini [20]). For these three methods, we set the LAION dataset [48] as the real class, and use the corresponding text descriptions to generate the fake images.

LDMs can be used to generate images in different ways. The standard practice is to use a text-prompt as input, and perform 200 steps of noise refinement (LDM 200). One can also generate an image with the help of guidance (LDM 200 w/CFG), or use fewer steps for faster sampling (LDM 100). Similarly, we also experiment with different variants of a pre-trained Glide model, which consists of two separate stages of noise refinement. The standard practice is to use 100 steps to get a low resolution image at 64×64, then use 27 steps to upsample the image to 256×256 in the next stage (Glide 100-27). We consider two other variants as well : Glide 50-27 and Glide 100-10 based on the number of refinement steps in the two stages. All generative models synthesize 256×256 resolution images.

5.2. Real-vs-Fake classification baselines

We compare with the following state-of-the-art baselines: (i) Training a classification network to give a real/fake decision for an image using binary cross-entropy loss [50]. The authors take a ResNet-50 [27] pre-trained on ImageNet, and finetune it on ProGAN's real/fake images (henceforth referred as trained deep network). (ii) We include another variant where we change the backbone to CLIP:ViT [24] (to match our approach) and train the network for the same task. (iii) Training a similar classification network on a patch level instead [10], where the authors propose to truncate either a ResNet [27] or Xception [14] (at Layer1 and Block2 respectively) so that a smaller receptive field is considered when making the decision. This method was primarily proposed for detecting generated *facial* images, but

we study whether the idea can be extended to detect more complex fake images. (iv) Training a classification network where input images are first converted into their corresponding co-occurrence matrices [35] (a technique shown to be effective in image steganalysis and forensics [16, 26]), conditioned on which the network predicts the real/fake class. (v) Training a classification network on the frequency spectrum of real/fake images [53], a space which the authors show as better in capturing and displaying the artifacts present in the GAN generated images. All training details can be found in the supplementary.

5.3. Evaluation metrics

We follow existing works [10, 25, 35, 50, 53] and report both average precision (AP) and classification accuracy. To compute accuracy for the baselines, we tune the classification threshold on the held-out training validation set of the available generative model. For example, when training a classifier on data associated with ProGAN, the threshold is chosen so that the accuracy on a held out set of ProGAN's real and fake images can be maximized. In addition, we also compute an upper-bound *oracle* accuracy for [50], where the classifier's threshold is calibrated directly on each test set separately. This is to gauge the best that the classifier can perform on each test set (details in supplementary).

6. Results

We start by comparing our approach to existing baselines in their ability to classify different types of real/fake images, and then study the different components of our approach.

6.1. Detecting fake images from unseen methods

Table 2 and Table 3 show the average precision (AP) and classification accuracy, respectively, of all methods (rows) in detecting fake images from different generative models (columns). For classification accuracy, the numbers shown are averaged over the real and fake classes for each generative model.[2] All methods have access to only ProGAN's data (except [53], which uses CycleGAN's data), either for training the classifier or for creating the NN feature bank.

As discussed in Sec. 3.2, the trained classifier baseline [50] distinguishes real from fakes with good accuracy for other GAN variants. However, the accuracy drops drastically (sometimes to nearly chance performance ∼50-55%; e.g., LDM variants) for images from most unseen generative models, where all types of fake images are classified mostly as real (please see Table C in the supplementary). Importantly, this behavior does not change even if we change the backbone to CLIP:ViT (the one used by our methods). This tells us that the issue highlighted in Fig. 2 affects deep neural networks in general, and not just ResNets. Performing classification on a patch-level [10], using co-occurence matrices [35], or using the frequency space [53] does not solve the issue either, where the classifier fails to have a consistent detection ability, sometimes even for methods within the same generative model family (e.g., GauGAN/BigGAN). Furthermore, even detecting real/fake patches in images from the same training domain (ProGAN) can be difficult in certain settings (Xception). This indicates that while learning to find patterns within small image regions might be sufficient when patches do not vary too much (e.g., facial images), it might not be sufficient when the domain of real and fake images becomes more complex (e.g., natural scenes).

Our approach, on the other hand, show a *drastically better generalization performance* in detecting real/fake images. We observe this first by considering models within the training domain, i.e., GANs, where our NN variants and linear probing achieve an average accuracy of ∼93% and ∼95% respectively, while the best performing baseline, trained deep networks - Blur+JPEG(0.5) achieves ∼85% (improvements of **+8-10%**). This discrepancy in performance becomes more pronounced when considering unseen methods such as diffusion (LDM+Guided+Glide) and autoregressive models (DALL-E), where our NN variants and linear probing achieve 82-84% average accuracy and ∼82% respectively compared to 53-58% by trained deep networks variants [50] (improvements of **+25-30%**). In terms of average precision, the best version of the trained deep network's AP is very high when tested on models from the same GAN family, 94.19 mAP, but drops when tested on unseen diffusion/autoregressive models, 75.51 mAP. Our NN variants and linear probing maintain a high AP both within the same (GAN) family domain, 96.36 and 99.31 mAP, and on unseen diffusion/autoregressive models, 90.58 and 95.00 mAP, resulting in an improvement of about **+15-20 mAP**. These improvements remain similar for our NN variants for voting pool size from k=1 to k=9, which shows that our method is not too sensitive to this hyperparameter.

In sum, these results clearly demonstrate the advantage of using the feature space of a frozen, pre-trained network that is *blind* to the downstream real/fake classification task.

6.2. Allowing the trained classifier to cheat

As described in Sec. 5.3, we experiment with an oracle version of the trained classifier baseline [50], where the threshold of the classifier is tuned directly on each *test set*. Even this flexibility, where the network essentially *cheats(!)*, does not make that classifier perform nearly as well as our approach, especially for models from unseen domains; for example, our nearest neighbor ($k = 9$) achieves an average classification accuracy of 84.25%, which is **7.99% higher than that of the oracle baseline** (76.26%). This shows that the issue with training neural networks for this task is not just the improper threshold at test time. In-

[2] See appendix which further breaks down the accuracies for real/fake.

Detection method	Variant	Generative Adversarial Networks						Deep fakes	Low level vision		Perceptual loss		Guided	LDM			Glide			DALL-E	Total
		Pro-GAN	Cycle-GAN	Big-GAN	Style-GAN	Gau-GAN	Star-GAN		SITD	SAN	CRN	IMLE		200 steps	200 w/ CFG	100 steps	100 27	50 27	100 10		mAP
Trained deep network [50]	Blur+JPEG (0.1)	**100.0**	93.47	84.5	**99.54**	89.49	98.15	89.02	73.75	59.47	98.24	98.4	73.72	70.62	71.0	70.54	80.65	84.91	82.07	70.59	83.58
	Blur+JPEG (0.5)	**100.0**	96.83	88.24	98.29	98.09	95.44	66.27	86.0	61.2	**98.94**	**99.52**	68.57	66.0	66.68	65.39	73.29	78.02	76.23	65.93	81.52
	ViT:CLIP (B+J 0.5)	99.98	93.32	83.63	88.14	92.81	84.62	67.23	**93.48**	55.21	88.75	96.22	55.74	52.52	54.51	52.2	56.64	61.13	56.64	62.74	73.44
Patch classifier [10]	ResNet50-Layer1	98.86	72.04	68.79	92.96	55.9	92.06	60.18	65.82	52.87	68.74	67.59	70.05	87.84	84.94	88.1	74.54	76.28	75.84	77.07	75.28
	Xception-Block2	80.88	72.84	71.66	85.75	65.99	69.25	76.55	76.19	76.34	74.52	68.52	75.03	87.1	86.72	86.4	85.37	83.73	78.38	75.67	77.73
Co-occurence [35]	-	99.74	80.95	50.61	98.63	53.11	67.99	59.14	68.98	60.42	73.06	87.21	70.20	91.21	89.02	92.39	89.32	88.35	82.79	80.96	78.11
Freq-spec [53]	CycleGAN	55.39	**100.0**	75.08	55.11	66.08	**100.0**	45.18	47.46	57.12	53.61	50.98	57.72	77.72	77.25	76.47	68.58	64.58	61.92	67.77	66.21
Ours	NN, $k = 1$	**100.0**	98.14	94.49	86.68	99.26	99.53	**93.09**	78.46	67.54	83.13	91.07	79.31	95.84	79.84	95.97	93.98	95.17	**96.05**	88.51	90.32
	NN, $k = 3$	**100.0**	98.13	94.46	86.67	99.25	99.53	93.03	78.54	67.54	83.13	91.06	79.26	95.81	79.78	95.94	93.94	95.13	94.60	88.47	90.22
	NN, $k = 5$	**100.0**	98.13	94.46	86.66	99.25	99.53	93.02	78.54	67.54	83.12	91.06	79.25	95.81	79.78	95.94	93.94	95.13	94.60	88.46	90.22
	NN, $k = 9$	**100.0**	98.13	94.46	86.66	99.25	99.53	91.67	78.54	67.54	83.12	91.06	79.24	95.81	79.77	95.93	93.93	95.12	94.59	88.45	90.14
	LC	**100.0**	99.46	**99.59**	97.24	**99.98**	99.60	82.45	61.32	**79.02**	96.72	99.00	**87.77**	**99.14**	**92.15**	**99.17**	**94.74**	**95.34**	94.57	**97.15**	**93.38**

Table 2. **Generalization results.** Average precision (AP) of different methods for detecting real/fake images. Models outside the GANs column can be considered as the generalizing domain. The improvements using the fixed feature backbone (Ours NN/LC) over the best performing baseline [50] is particularly noticeable when evaluating on unseen generative models, where our best performing method has significant gains over the best performing baseline: +9.8 mAP overall and +19.49 mAP across unseen diffusion & autoregressive models.

Detection method	Variant	Generative Adversarial Networks						Deep fakes	Low level vision		Perceptual loss		Guided	LDM			Glide			DALL-E	Total
		Pro-GAN	Cycle-GAN	Big-GAN	Style-GAN	Gau-GAN	Star-GAN		SITD	SAN	CRN	IMLE		200 steps	200 w/ CFG	100 steps	100 27	50 27	100 10		Avg. acc
Trained deep network [50]	Blur+JPEG (0.1)	99.99	85.20	70.20	85.7	78.95	91.7	53.47	66.67	48.69	86.31	86.26	60.07	54.03	54.96	54.14	60.78	63.8	65.66	55.58	69.58
	Blur+JPEG (0.5)	**100.0**	80.77	58.98	69.24	79.25	80.94	51.06	56.94	47.73	87.58	94.07	51.90	51.33	51.93	51.28	54.43	55.97	54.36	52.26	64.73
	Oracle* (B+J 0.5)	**100.0**	90.88	82.40	**93.11**	93.52	87.27	62.48	**76.67**	57.04	**95.28**	**96.93**	65.20	63.15	62.39	61.50	65.36	69.52	66.18	60.10	76.26
	ViT:CLIP (B+J 0.5)	98.94	78.80	60.62	60.56	66.82	62.31	52.28	65.28	47.97	64.09	79.54	50.66	50.74	51.04	50.76	52.15	53.07	52.06	53.18	60.57
Patch classifier [10]	ResNet50-Layer1	94.38	67.38	64.62	82.26	57.19	80.29	55.32	64.59	51.24	54.29	55.11	65.14	79.09	**76.17**	79.36	67.06	68.55	68.04	69.44	68.39
	Xception-Block2	75.03	68.97	68.47	79.16	64.23	63.94	75.54	75.14	**75.28**	72.33	55.3	67.41	76.5	76.1	75.77	74.81	73.28	68.52	67.91	71.24
Co-occurence [35]	-	97.70	63.15	53.75	92.50	51.1	54.7	57.1	63.06	55.85	65.65	65.80	60.50	70.7	70.55	71.00	70.25	69.60	69.90	67.55	66.86
Freq-spec [53]	CycleGAN	49.90	**99.90**	50.50	49.90	50.30	**99.70**	50.10	50.00	48.00	50.60	50.10	50.90	50.40	50.40	50.30	51.70	51.40	50.40	50.00	55.45
Ours	NN, $k = 1$	99.58	94.70	86.95	80.24	96.67	98.84	80.9	71.0	56.0	66.3	76.5	68.76	89.56	68.99	89.51	86.44	88.02	87.27	77.52	82.30
	NN, $k = 3$	99.58	95.04	87.63	80.55	96.94	98.77	83.05	71.5	59.5	66.69	76.87	70.02	90.37	70.17	90.57	87.84	89.34	88.78	79.29	83.28
	NN, $k = 5$	99.60	94.32	88.23	80.60	97.00	98.90	83.85	71.5	60.0	67.04	78.02	70.55	90.89	70.97	91.01	88.42	90.07	89.60	80.19	83.72
	NN, $k = 9$	99.54	93.49	88.63	80.75	97.11	98.97	**84.5**	71.5	61.0	69.27	79.21	**71.06**	91.29	72.02	91.29	**89.05**	**90.67**	**90.08**	81.47	**84.25**
	LC	**100.0**	98.50	**94.50**	82.00	**99.50**	97.00	66.60	63.00	57.50	59.5	72.00	70.03	**94.19**	73.76	**94.36**	79.07	79.85	78.14	**86.78**	81.38

Table 3. **Generalization results.** Analogous result of Table 2, where we use classification accuracy (averaged over real and fake images) to compare the methods. Oracle with * indicates that the method uses the test set to calibrate the confidence threshold. The fixed feature backbone (Ours NN/LC) has a significant gain in accuracy (+25-30% over the baselines) when testing on unseen generative model families.

stead, the trained network fundamentally cannot do much other than to look for a certain set of fake patterns; and in their absence, has issues looking for features pertaining to the real distribution. And that is where the feature space of a *model not trained on this task* has its advantages; even when those fake features are absent, there will still be other features useful for classification, which were not learned to be ruled out during the real-vs-fake training process.

6.3. Effect of network backbone

So far, we have seen the surprisingly good generalizability of nearest neighbor / linear probing using CLIP:ViT-L/14's feature space. In this section, we study what happens if the backbone architecture or pre-training dataset is changed. We experiment with our linear classification variant, and consider the following settings: (i) CLIP:ViT-L/14, (ii) CLIP:ResNet-50, (iii) ImageNet:ResNet-50, and (iv) ImageNet:ViT-B/16. For each, we again use ProGAN's real/fake image data as the training data.

Fig. 5 shows the accuracy of these variants on the same models. The key takeaway is that both the network architecture as well as the dataset on which it was trained on play a crucial role in determining the effectiveness for fake image detection. Visual encoders pre-trained as part of the CLIP system fare better compared to those pre-trained on ImageNet. This could be because CLIP's visual encoder gets to see much more diversity of images, thereby exposing it to a much bigger *real distribution* than a model trained on ImageNet. Within CLIP, ViT-L/14 performs better than ResNet-50, which could partly be attributed to its bigger architecture and global receptive field of the attention layers.

We also provide a visual analysis of the pre-trained distributions. Using each of the four model's feature banks consisting of the same real and fake images from ProGAN, we plot four t-SNE figures and color code the resulting 2-D points using binary (real/fake) labels in Fig. 7. CLIP:ViT-L/14's space best separates the real (red) and fake (blue) features, followed by CLIP:ResNet-50. ImageNet:ResNet-50 and ImageNet:ViT-B/16 do not seem to have any proper

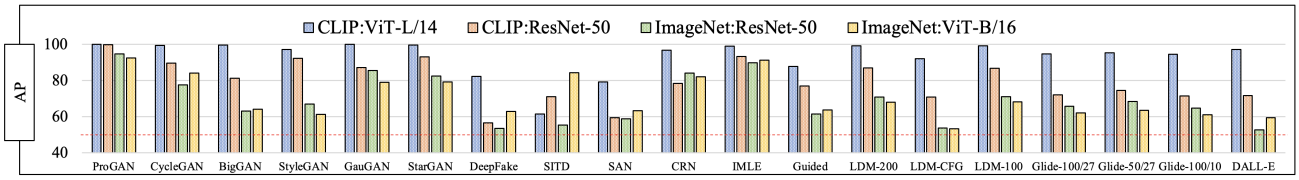

Figure 5. **Ablation on the network architecture and pre-training dataset.** A network trained on the task of CLIP is better equipped at separating fake images from real, compared to networks trained on ImageNet classification. The red dotted line depicts chance performance.

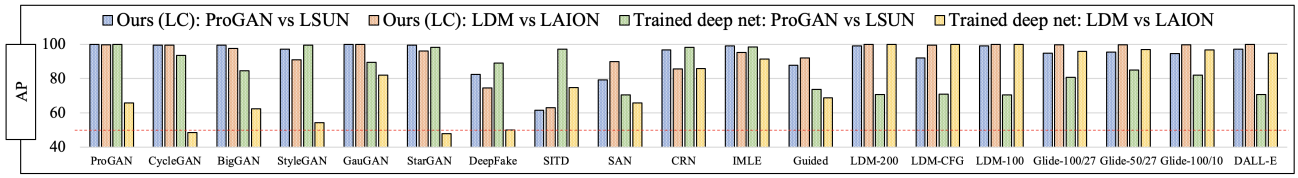

Figure 6. **Average precision of methods with respect to training data.** Both our linear classifier on CLIP:ViT's features and the baseline trained deep network [50] are given access to two different types of training data: (i) $\mathcal{R}$ = LSUN [51] and $\mathcal{F}$ = ProGAN [28], (ii) $\mathcal{R}$ = LAION [48] and $\mathcal{F}$ = LDM [46]. Irrespective of the training data source, our linear classifier preserves its ability to generalize well on images from other unseen generative model families, which is not the case for the baseline trained deep network.

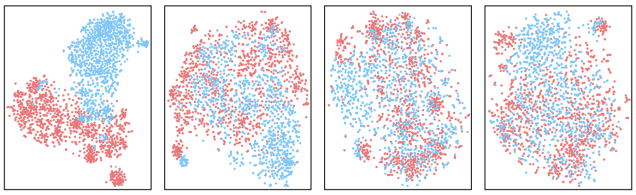

Figure 7. t-SNE visualization of real (red) and fake (blue) images using the feature space of different image encoders.

structure in separating the two classes, suggesting that the pre-training data matters more than the architecture.

6.4. Effect of training data source

So far, we have used ProGAN as the source of training data. We next repeat the evaluation setup in Table 2 using a pre-trained LDM [46] as the source instead. The real class consists of images from LAION dataset [48]. Fake images are generated using an LDM 200-step variant using text prompts from the corresponding real images. In total, the dataset consists of 400k real and 400k fake images.

Fig. 6 (top) compares our resulting linear classifier to the one created using ProGAN's dataset. Similar to what we have seen so far, access to only LDM's dataset also enables the model to achieve good generalizability. For example, our model can detect images from GAN's domain (now an unseen generative model), with an average of 97.32 mAP. In contrast, the trained deep network (Fig. 6 bottom) performs well only when the target model is from the same generative model family, and fails to generalize in detecting images from GAN variants, 60.17 mAP; i.e., the improvement made by our method for the unseen GAN domain is **+37.16 mAP**. In summary, with our linear classifier, one can start with ProGAN's data and detect LDM's fake images, or vice versa. This is encouraging because it tells us that, *so far*, with all the advancements in generative models, there is still a hidden link which connects various fake images.

7. Conclusion and Discussion

We studied the problem associated with training neural networks to detect fake images. The analysis paved the way for our simple fix to the problem: using an informative feature space *not trained* for real-vs-fake classification. Performing nearest neighbor / linear probing in this space results in a significantly better generalization ability of detecting fake images, particularly from newer methods like diffusion/autoregressive models. As mentioned in Sec. 6.4, these results indicate that even today there is something common between the fake images generated from a GAN and those from a diffusion model. However, what that similarity is remains an open question. And while having a better understanding of that question will be helpful in designing even better fake image detectors, we believe that the generalization benefits of our proposed solutions should warrant them as strong baselines in this line of work.

8. Acknowledgement

This work was supported in part by NSF CAREER IIS2150012, Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2022-00187238, Development of Large Korean Language Model Technology for Efficient Pre-training), and Adobe Data Science Research Award.

References

[1] Adjust and exaggerate facial features. https://helpx.adobe.com/photoshop/how-to/face-awareliquify.html.

[2] Deepfacelab. https://github.com/iperov/deepfacelab.

[3] Dfaker. https://github.com/dfaker/df.

[4] Faceswap. https://faceswap.dev/.

[5] Shruti Agarwal and Hany Farid. Photo forensics from jpeg dimples. In *IEEE Workshop on Information Forensics and Security*, 2017.

[6] Vishal Asnani, Xi Yin, Tal Hassner, Sijia Liu, and Xiaoming Liu. Proactive image manipulation detection. In *CVPR*, 2022.

[7] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv*, 2018.

[8] Mu Cai and Yixuan Li. Out-of-distribution detection via frequency-regularized generative models. In *WACV*, 2023.

[9] Mu Cai, Hong Zhang, Huijuan Huang, Qichuan Geng, Yixuan Li, and Gao Huang. Frequency domain image translation: More photo-realistic, better identity-preserving. In *ICCV*, pages 13930–13940, 2021.

[10] Lucy Chai, David Bau, Ser-Nam Lim, and Phillip Isola. What makes fake images detectable? understanding properties that generalize. In *ECCV*, 2020.

[11] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *CVPR*, 2018.

[12] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017.

[13] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018.

[14] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *arXiv*, 2017.

[15] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Splicebuster: A new blind image splicing detector. In *IEEE International Workshop on Information Forensics and Security*, 2015.

[16] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. In *arXiv*, 2017.

[17] Davide Cozzolino, Justus Thies, Rossler Andreas, Riess Christian, Nießner Matthias, and Luisa Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. In *arXiv*, 2019.

[18] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Zhang Lei. Second-order attention network for single image super-resolution. In *CVPR*, 2019.

[19] Cozzolino Davide, Thies Justus, Andreas Rossler, Matthias Nießner, and Luisa Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. In *arXiv*, 2019.

[20] Boris Dayma, Suraj Patil, Pedro Cuenca, Khalid Saifullah, Tanishq Abraham, Phúc Lê Khac, Luke Melas, and Ritobrata Ghosh. Dall·e mini, 2021.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[22] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021.

[23] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021.

[24] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *arXiv*, 2020.

[25] Joel Frank, Thorsten Eisenhofer, Lea Schonherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging frequency analysis for deep fake image recognition. In *ICML*, 2020.

[26] Jessica Fridrich and Jan Kodovsky. Rich models for steganalysis of digital images. In *IEEE Transactions on Information Forensics and Security*, 2012.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[28] Terro Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018.

[29] Terro Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[30] Ke Li, Tianhao Zhang, and Jitendra Malik. Diverse image synthesis from semantic layouts via conditional imle. In *ICCV*, 2019.

[31] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation. *CVPR*, 2023.

[32] Yuheng Li, Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Mixnmatch: Multifactor disentanglement and encoding for conditional image generation. In *CVPR*, 2020.

[33] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

[34] Francesco Marra, Diego Gragnaniello, Cozzolino Davide, and Luisa and Verdoliva. Detection of gan-generated fake images over social networks. In *IEEE Conference on Multimedia Information Processing and Retrieval*, 2018.

[35] Lakshmanan Natraj, Tajuddin Manhar Mohammed, Shivkumar Chandrasekaran, Arjuna Flenner, Amit K. Roy-Chowdhuri, and B.S. Manjunath. Detecting gan generated fake images using co-occurrence matrices. In *Electronic imaging*, 2019.

[36] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *ICML*, 2022.

[37] James F. O'Brien and Hany Farid. Exposing photo manipulation with inconsistent reflections. In *ACM Transactions on Graphics*, 2012.

[38] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[39] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei A. Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. In *Advances in Neural Information Processing Systems*, 2020.

[40] Alin C. Popescu and Hany Farid. Exposing digital forgeries by detecting traces of resampling. In *IEEE Transactions on signal processing*, 2005.

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[42] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. In *arXiv*, 2022.

[43] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021.

[44] Yuan Rao and Jiangqun Ni. A deep learning approach to detection of splicing and copy-move forgeries in images. In *IEEE International Workshop on Information Forensics and Security*, 2016.

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[46] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[47] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. FaceForensics++: Learning to detect manipulated facial images. In *International Conference on Computer Vision*, 2019.

[48] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. In *Data Centric AI NeurIPS Workshop 2021*, 2021.

[49] Sheng-Yu Wang, Oliver Wang, Andrew Owens, Richard Zhang, and Alexei A Efros. Detecting photoshopped faces by scripting photoshop. In *ICCV*, 2019.

[50] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot...for now. In *CVPR*, 2020.

[51] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. 2015.

[52] Ning Yu, Larry S. Davis, and Mario Fritz. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *ICCV*, 2019.

[53] Zu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In *WIFS*, 2019.

[54] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.