The Ultimate Combo: Boosting Adversarial Example Transferability by Composing Data Augmentations

Zebin Yun
zebinyun@mail.tau.ac.il
Tel Aviv University
Tel Aviv, Israel

Achi-Or Weingarten
achi.wgn@gmail.com
Weizmann Institute of Science
Rehovot, Israel

Eyal Ronen*
eyalronen@tauex.tau.ac.il
Tel Aviv University
Tel Aviv, Israel

Mahmood Sharif*
mahmoods@tauex.tau.ac.il
Tel Aviv University
Tel Aviv, Israel

Abstract

To help adversarial examples generalize from surrogate machine-learning (ML) models to targets, certain transferability-based black-box evasion attacks incorporate data augmentations (e.g., random resizing). Yet, prior work has explored limited augmentations and their composition. To fill the gap, we systematically studied how data augmentation affects transferability. Specifically, we explored 46 augmentation techniques originally proposed to help ML models generalize to unseen benign samples, and assessed how they impact transferability, when applied individually or composed. Performing exhaustive search on a small subset of augmentation techniques and genetic search on all techniques, we identified augmentation combinations that help promote transferability. Extensive experiments with the ImageNet and CIFAR-10 datasets and 18 models showed that simple color-space augmentations (e.g., color to greyscale) attain high transferability when combined with standard augmentations. Furthermore, we discovered that composing augmentations impacts transferability mostly monotonically (i.e., more augmentations $\rightarrow \geq$transferability). We also found that the best composition significantly outperformed the state of the art (e.g., 91.8% vs. $\leq$82.5% average transferability to adversarially trained targets on ImageNet). Lastly, our theoretical analysis, backed by empirical evidence, intuitively explains why certain augmentations promote transferability.

CCS Concepts

• **Computing methodologies** → **Neural networks**; • **Security and privacy** → *Software and application security*.

Keywords

Adversarial Examples, Transferability, Neural Networks

*Corresponding authors.

This work is licensed under a Creative Commons Attribution International 4.0 License.

AISec '24, October 14–18, 2024, Salt Lake City, UT, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1228-9/24/10
https://doi.org/10.1145/3689932.3694769

ACM Reference Format:
Zebin Yun, Achi-Or Weingarten, Eyal Ronen, and Mahmood Sharif. 2024. The Ultimate Combo: Boosting Adversarial Example Transferability by Composing Data Augmentations. In *Proceedings of the 2024 Workshop on Artificial Intelligence and Security (AISec '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3689932.3694769

1 Introduction

Adversarial examples (AEs)—variants of benign inputs minimally perturbed to induce misclassification at test time—have emerged as a profound challenge to machine learning (ML) [1, 40], calling its use in security- and safety-critical systems into question (e.g., [10]). Many attacks have been proposed to generate AEs in white-box settings, where adversaries are familiar with all the particularities of the attacked model [30]. By contrast, black-box attacks enable evaluating the vulnerability of ML in realistic settings, without access to the model [30].

Attacks exploiting the transferability-property of AEs [40] have received special attention. Namely, as AEs produced against one model are often misclassified by others, transferability-based attacks produce AEs against surrogate (a.k.a. substitute) white-box models to mislead black-box ones. To measure the risk of AEs in black-box settings accurately, researchers have proposed varied methods to enhance transferability (e.g., [25–27]). Notably, attacks using data augmentation, such as translations [8] and scaling of pixel values [26], as a means to improve the generalizability of AEs across models have accomplished state-of-the-art transferability rates. Still, previous transferability-based attacks have studied only six augmentation methods (see §3.1), out of many proposed in the data-augmentation literature, primarily for reducing model overfitting [35]. Hence, the extent to which different data-augmentation types boost transferability, either individually or when combined, remains largely unknown.

To fill the gap, we conducted a systematic study of how augmentation methods influence transferability. Specifically, alongside techniques considered in previous work, we studied how 46 augmentation techniques pertaining to seven categories impact transferability when applied individually or composed (§3). Integrating augmentation methods into attacks via a flexible framework (Alg. 1), we searched for augmentation-combinations that can help boost transferability via inefficient but optimal exhaustive search on a small subset of augmentations and efficient, near-optimal genetic

search on a search space containing all augmentations (§3.4). We conducted extensive experiments using an ImageNet-compatible dataset, CIFAR-10 [22], and 18 models, and measured transferability in diverse settings, including with and without defenses (§5–6). Furthermore, by studying the impact of augmentations on model gradients, we offer a theoretical explanation for why certain augmentations can promote transferability (§4) that we later support with empirical results (§6.7).

In a nutshell, we make the following contributions:

- We find that simple color-space augmentations exhibit performance commensurate with that of state-of-the-art techniques. Notably, they surpass state-of-the-art techniques in terms of transferability to adversarial training models, while simultaneously offering a reduction in running time costs (§6.1 and §6.6).
- We propose parallel composition to integrate a large number of augmentations into attacks (§3.3) and find it boosts transferability compared to composition approaches previously used (§6.5). Further, with parallel composition, we discover that transferability has a mostly monotonic relationship with the new data-augmentation techniques we introduce and ones used in prior work (§6.2).
- We show that attacks integrating the best augmentation combinations discovered by exhaustive and genetic search ($\text{UltComb}_{\text{Base}}$ and $\text{UltComb}_{\text{Gen}}$, respectively) outperform state-of-the-art attacks by a large margin (§6.4).
- We theoretically demonstrate that augmentations can smoothen surrogate models' gradients, which in turn can improve transferability (§4), and empirically back the theory (§6.7).

2 Background and Related Work

Evasion Attacks Many evasion attacks assume adversaries have white-box access to models—i.e., adversaries know models' architectures and weights (e.g., [3, 12, 40]). These typically leverage first- or second-order optimizations to generate AEs models would misclassify. For example, given an input x of class y, model weights θ, and a loss function J, the Fast Gradient Sign method (FGSM) of [12], crafts an AE $\hat{x}$ using the loss gradients $\nabla_x J(x, y, \theta)$:

$$\hat{x} = x + \epsilon * \text{sign}(\nabla_x J(x, y, \theta))$$

where sign$(\cdot)$ maps real numbers to -1, 0, or 1, depending on their sign. Following FGSM, researchers proposed numerous advanced attacks. Notably, iterative FGSM (I-FGSM) performs multiple update steps to $\hat{x}$ to evade models [24]:

$$\hat{x}_{t+1} = \text{Proj}_x^{\epsilon}\left(\hat{x}_t + \alpha \cdot \text{sign}\left(\nabla_x J\left(\hat{x}_t, y, \theta\right)\right)\right)$$

where $\text{Proj}_x^{\epsilon}(\cdot)$ projects the perturbation into ℓ_∞-norm ϵ-ball centered at x, α is the step size, and $\hat{x}_0 = x$. In this work, we study attacks based on I-FGSM.

In practice, adversaries often lack white-box access to victim models. Hence, researchers studied black-box attacks in which adversaries may only query models. Certain attack types, such as score- and boundary-based attacks perform multiple queries, often around several thousands, to produce AEs (e.g., [2, 19]). By contrast, attacks leveraging *transferability* (e.g., [12, 40]) avoid querying victim models, and use surrogate white-box models to create AEs that are likely misclassified by other black-box ones.

Enhancing transferability is an active research area. Some methods integrate momentum into attacks such as MI-FGSM to avoid surrogate-specific optima and saddle points that may hinder transferability (e.g., [7, 45, 47]). Others employ specialized losses, such as reducing the variance of intermediate activations [18] or the mean loss of model ensembles [27], to enhance transferability. Lastly, a prominent family of attacks leverages data augmentation to enhance AEs' generalizability between models. For instance, Dong et al. boosted transferability by integrating random translations into I-FGSM [8]. Evasion attacks incorporating data augmentation attain state-of-the-art transferability rates [26, 45, 46]. Nonetheless, prior work has only considered a restricted set of four augmentation methods for boosting transferability (see §3.1). By contrast, we investigate augmentations' role at enhancing transferability more systematically, by exploring how a more comprehensive set of augmentation types and their compositions affect transferability. Some efforts explored how augmentation methods used during training affect transferability [28, 59]. Unlike these, we study augmentations' role in boosting transferability when incorporated into attacks.

Defenses Various defenses were proposed to mitigate evasion attacks. Adversarial training—a procedure integrating correctly labeled AEs in training—is of the most effective methods for enhancing adversarial robustness (e.g., [12, 43]). Other defense methods sanitize inputs prior to classification (e.g., [14]); attempt to detect attacks (see [42]); or seek to certify robustness in ϵ-balls around inputs (e.g., [4, 33]). Following standard practices in the literature [46], we evaluate transferability-based attacks against a representative set of these defenses.

3 Data Augmentations for Transferability

Data augmentation is traditionally used in training, to reduce overfitting and improve generalizability [35]. Inspired by this use, transferability-based attacks adopted data augmentation to limit overfitting to surrogate models and produce AEs likely to generalize and be misclassified by victim models. Alg. 1 presents a general framework for integrating data augmentation into I-FGSM with momentum (MI-FGSM) [7]. In the framework, a method $D(\cdot)$ augments the attack with m variants of the estimated AE at each iteration. Consequently, the adversarial perturbation found by the attack increases the expected loss over transformed counterparts of the benign sample x (i.e., the distribution set by $D(\cdot)$ given x).

The framework in Alg. 1 is flexible, and can admit any augmentation method. We use it to describe previous attacks employing augmentations and to systematically explore new ones. Next, we detail previous attacks, describe augmentation methods we adopt for the first time to enhance transferability, and explain how these can be combined for best performance.

3.1 Previous Attacks Leveraging Augmentations

Prior work explored the following augmentation methods (to set $D(\cdot)$ in Alg. 1).

Algorithm 1 MI-FGSM with augs.

Input: Sample x; label y; loss $J(\cdot)$; model params. θ; # iters. T; momentum μ; norm-bound ϵ; method $D(\cdot)$ producing m augmented samples; step size α.
$\alpha = \epsilon/T$
$\hat{x}_0 = x$
$g_0 = 0$
for $t = 0$ to $T-1$ **do**
  $\bar{g}_{t+1} = \frac{1}{m}\sum_{i=0}^{m-1} \nabla_x \left(J\left(D(\hat{x}_t)_i, y, \theta\right)\right)$
  $g_{t+1} = \mu \cdot g_t + \frac{\bar{g}_{t+1}}{\|\bar{g}_{t+1}\|_1}$
  $\hat{x}_{t+1} = \text{Proj}_x^{\epsilon}\left(\hat{x}_t + \alpha \cdot \text{sign}\left(g_{t+1}\right)\right)$
end for
return $\hat{x} = \hat{x}_T$

Translations Using random translations of inputs, Dong et al. proposed a translation-invariant attack to promote transferability [8]. They also offered an optimization to reduce the attack's time and space complexity by convolving the model's gradients (w.r.t. non-translated inputs) with a Gaussian kernel. While we use this optimization in the implementation in the interest of efficiency, we highlight that Alg. 1 can capture the attack.

Diverse Inputs Xie et al. proposed a size-invariant attack. Their augmentation randomly crops $\hat{x}_t$ and resizes crops per the model's input dimensionality [51].

Scaling Pixels Lin et al. showed that adversarial perturbations invariant to scaling pixel values transfer with higher success between deep neural networks (DNNs) [26]. In their case, $D(\cdot)$ produces m samples such that $D(x)_i = \frac{x}{2^i}$ for $i \in \{0, 1, ..., m-1\}$, where m=5 by default.

Admix Wang et al. assumed that the adversary has a gallery of images from different classes and adopted augmentations similar to *MixUp* [46, 57]. For each sample x' from the gallery (typically selected at random from the batch), Admix augments attacks with m (typically set to 5) samples, such that $D(x, x')_i = \frac{1}{2^i} \cdot (\hat{x}_t + \eta \cdot x')$, where $i \in \{0, 1, ..., m-1\}$, and $\eta \in [0, 1]$ is set to 0.2 by default. Notably, Admix degenerates to *scaling pixels* when $\eta = 0$.

Uniform Noise (UN) Li et al. added uniform noise to samples to produce uniform-noise-invariant attacks with enhanced transferability [25].

Dropping Patches (DP) Li et al. and Wang et al. divided perturbations into (16×16) grids and randomly sampled parts of the grid to drop [25, 48].

Leading transferability-based attacks compose *(1)* diverse inputs, scaling, and translations (Lin et al.'s DST attack [26], and Wang et al.'s VMI-DST attack that also tunes the gradients' variance [45]); or *(2)* Admix, diverse inputs, and translation (Wang et al.'s Admix-DT attack [46]); or *(3)* uniform noise, dropping patches, diverse inputs, and translations (Li et at.'s UNDP-DT [25]). We consider these attacks as baselines in our experiments (§5–6).

3.2 New Augmentations for Boosting Transferability

While prior work studied the effect of some data-augmentation methods' effect on transferability, a substantially wider range of data-augmentation methods exist. Yet, the impact of these on transferability remains unknown. To fill the gap, we examined Shorten et al.'s survey on data augmentation [35] as well as two prominent image-augmentation libraries [20, 32] for reducing overfitting in deep learning to identify a comprehensive set of augmentation methods that may boost transferability. Overall, we identified 46 representative methods of seven categories to evaluate. We present a representative subset of augmentations per category below; App. A lists the remaining augmentations.

Color-space Transformations Potentially the simplest augmentation types are those applied in color-space. Given images represented as three-channel tensors, methods in this category manipulate pixel values only based on information encoded in the tensors. We evaluate 12 color-space transformations. Among them, we consider *color jitter (CJ)*, which applies random color manipulations [49]. Specifically, we consider random adjustments of pixel values within a pre-defined ranges of hue, contrast, saturation, and brightness. Additionally, we consider *greyscale (GS)* augmentations. This simple augmentation converts images into greyscale (replicating it three times to obtain an RGB representation). Mathematically, the conversion is calculated by $\omega_R \cdot x^R + \omega_G \cdot x^G + \omega_B \cdot x^B$, where x^R, x^G, and x^B, correspond to the RGB channels, respectively, and ω_R, ω_G, and ω_B, all $\in [0, 1]$, denote the channel weights, and sum up to 1.

Random Deletion Different from DP, which drops random regions from *perturbations*, we consider random deletions of *input samples'* regions, replacing them with others values. For example, inspired by dropout regularization, *random erasing (RE)* helps ML models focus on descriptive features of images and promote robustness to occlusions [60]. To do so, randomly selected rectangular regions in images are replaced by masks composed of random pixel values. Similarly to RE, *CutOut* masks out regions of inputs to improve DNNs' accuracy [6]. The main difference from RE is that CutOut uses fixed masking values, and may perform less aggressive masking when selected regions lie outside the image.

Kernel Filters Convolving images with filters of different types can produce certain effects, such as blurring (via Gaussian kernels), sharpening (via edge filters), or edge enhancement. We study the effect of 13 filters on transferability, including edge-enhancement via *sharpening (Sharp)*.

Mixing Images Some augmentation methods (e.g., Admix) fuse images together. We consider *CutMix*, which replaces a region within one image with a region from another image picked from a gallery [55].

Style Transfer Certain augmentations change the image style. We study six of these in this work. Among the six, we also consider augmentations using *neural transfer (NeuTrans)* to preserve image semantics while changing their style. Specifically, we use [11]'s generative model to transfer image styles to that of Picasso's 1907 self-portrait.

Meta-learning-inspired Augmentations Meta-learning is a subfield of ML studying how ML algorithms can optimize other learning algorithms [16]. In the context of data augmentation, algorithms such as *AutoAugment* have been proposed to train controllers to select an appropriate augmentation method to avoid overfitting [5]. We use the pre-trained AutoAugment controller, encoded as a recurrent neural network, to select augmentation methods and their magnitude from a set of pre-defined augmentation methods.

Spatial Transformations Augmentations performing spatial transformations alter the locations of pixels in the image. Random translation and resizing (§3.1) belong to this category. We consider seven additional augmentations that perform spatial transformation, including random rotations, affine transformations, and horizontal flipping.

3.3 Composing Augmentations

There exist two ways to compose data-augmentation methods in attacks, namely: *parallel* and *serial* composition. In parallel composition, each augmentation method is applied independently on the input, and their outputs are aggregated by taking their union as $D(\cdot)$'s output to augment attacks. By contrast, serial composition applies augmentations sequentially, one after the other, where the first method operates on the original sample, and each subsequent augmentation operates on its predecessor's outputs. Consequently, serial composition leads to an *exponential* growth in the number of samples, while parallel composition leads to a *linear* growth. All baselines we consider in this work [25, 26, 45, 46] employ serial composition, as this is the only composition method previously used. Compared to prior work, we consider a substantially larger number of augmentation methods, which may lead to prohibitive memory and time requirements in the case of serial composition. Additionally, because the order of applying certain augmentations matters (e.g., GS then CutMix leads to different outcome that CutMix followed by GS), exploring a meaningful number of serial compositions (e.g., out of $\geq$46! possibilities) becomes virtually impossible. Moreover, serial composition of images may distort images drastically, thus degrading transferability, as we find (§6.5). Accordingly, we mainly consider parallel composition between data-augmentation methods. We *only* serially compose translations, scaling, and diverse inputs, for consistency with prior work (e.g., [46]).

3.4 Discovering Effective Compositions

We use two approaches to discover augmentation compositions that promote transferability. As a simple, yet effective method, we employ exhaustive search on a small set of augmentations. We note that, while Li et al. considered a similar approach [25], they only *(1)* evaluated augmentations previously established as conducive transferability, as opposed the the new augmentations we incorporate; and *(2)* tested serial compositions of augmentations, which we find to perform worse than parallel composition (§6.5). Exhaustive search enabled us to uncover a roughly monotonic relationship between augmentations and transferability (§6.2). Additionally, we apply genetic search over the full set of augmentations. Here, informed by the roughly monotonic relationship between augmentations and transferability, we bias the search hyperparameters toward including augmentations within combinations to speed up convergence toward combinations boosting transferability.

Exhaustive Search A straightforward approach to find a set of augmentations that promote transferability when combined is to exhaustively evaluate all possible combinations. On the one hand, this approach is guaranteed to find the combination that advances transferability the most, and enables us to precisely characterize the relationship between augmentations and transferability—e.g., whether transferability is monotonic in the number of augmentations included. However, it would be infeasible to exhaustively test all possible combinations included in the power set of all augmentations we consider in this work. Hence, as a middle ground, we evaluate exhaustive search with a representative, well-performing augmentation method per data-augmentation category.

Genetic Search As an efficient alternative for exhaustive search, we employ genetic search—a commonly used optimization technique suitable for large, discrete search spaces [21]. Genetic search algorithms seek to simulate biological evolution processes to find near-optimal solutions. Instead of evaluating the entire search space, genetic search begins with an initial population sampled from the search space. Subsequently, the algorithm selects the fittest individuals from the population and applies crossover and mutation actions between them to produce a generation that should score higher on the fitness objective one aims to optimize. Once the fitness value of the best-performing individuals converges or a maximum number of iteration is reached, the search ends and the best performing individual is returned.

In this work, we seek to search over a population of augmentation combinations to maximize transferability (i.e., the fitness function). To this end, starting with a population of containing several combinations, we evaluate the transferability rate achievable via each combination. Next, to produce the next generation, we select the fittest combinations with the highest transferability rates at random, with replacement, to act as ancestors, while maintaining the population size. The probability for selecting each combination is equal to the ratio between its fitness and the total fitness of all combinations in the population. Then, for each selected combination, with probability p_{cross} we perform crossover with another combination chosen at random from combinations that passed selection. If no crossover occurs, we pass the combination as is to the next generation after performing mutation. In crossover, we select (resp. exclude) an augmentation if it is included (resp. excluded) in both ancestors, and select whether to include the augmentation at random when it is included in one ancestor but excluded in the other. Subsequently, we perform mutations, removing (resp. inserting) and augmentation that is included (resp. removed) with probability p_{mutate}. We stop the process after n_{gen} search iterations.

As exhaustive search demonstrated a roughly monotonic relationship between augmentations and transferability (§6.2), we biased the genetic search toward including augmentations to speed up convergence to highly-performing combinations. More precisely, we selected an initial population with probability of p_{aug} >50% to include each augmentation in a combination. Thus, augmentation combinations in the initial and subsequent populations had higher likelihood to include augmentations than exclude them.

Genetic search is not guaranteed to converge to the optimal combinations as long as we have not covered the entire search space. Still, we find that, for small search spaces, genetic search quickly discovers combinations that attain roughly as high transferability rates as the best combination found by exhaustive search (§6.4). After identifying the genetic-search hyperparameters that work well for small search spaces, we apply genetic search to the large search space containing all augmentations. Doing so enables us to identify a combination of augmentations attaining state-of-the-art transferability rates when composed (§6.3).

4 Theoretical Analysis of Augmentations

Before turning to empirical evaluations, we theoretically analyze why augmentations can boost transferability. Specifically, we explore how augmentations affect the gradients attacks leverage to improve our understanding of how they help AEs generalize from surrogates to targets. Our analysis shows that certain augmentations smoothen the model gradients (i.e., lead gradients to be more consistent across attack iterations). Intuitively, in return, this may limit the effect of surrogate peculiarities (e.g., surrogate-specific changes in the classification boundaries) on the attack, and increase AEs' generalization. By contrast, when gradients are sensitive to surrogates' peculiarities, they are likely to change drastically within small regions of input domains (e.g., across attack iterations), hindering transferability to target models. Our experiments (§6.7) empirically support the theoretical analysis and intuition.

Adapting proof techniques from randomized smoothing [4, 33], we show that Gaussian noise helps smoothen gradients. Subsequently, we extend the result to a more general set of augmentation methods, showing that gradients can be smoothened by augmenting attacks with additive noise sampled from smooth distributions—i.e., distributions that do not exhibit sharp changes in their density within small regions.

Gaussian-Noise Augmentations Prior work has shown that augmenting inference with Gaussian noise, a technique known as randomized smoothing, leads to estimating a smooth function with a bounded Lipschitz constant (i.e., maximum rate of change in the region the function is defined) [4, 33]. Differently from prior work, instead of augmenting inference, ours augments the derivative (i.e., gradient estimates). We demonstrate that when using Gaussian noise for data augmentation, the derivative becomes smooth. More formally, we show that the derivative's Lipschitz constant becomes bounded. Moreover, we demonstrate that the smoother, less peaky, the Gaussian distribution becomes, the smaller is the derivative's Lipschitz constant and the smoother is the derivative.

For a sample and label pair (x, y), we denote the loss derivative as $f(x) = \nabla_x J(x, y, \theta)$. Let I be the Lipschitz constant of $J(x, y, \theta)$ (i.e., the surrogate's loss). Note that I is typically bounded for DNNs and can be approximated via numerical analysis techniques [44]. Our first theoretical result shows that $\hat{f}$, a variant of f augmented with noise sampled from a zero-mean Gaussian distribution with an identity covariance matrix E (i.e., noise drawn from $\mathcal{N}(0, E)$), is smooth, with a Lipschitz constant of $I \cdot \sqrt{\frac{2}{\pi}}$.

Theorem 4.1. *$\hat{f}$, attained by augmenting $\nabla_x J(x, y, \theta)$ with noise drawn from $\mathcal{N}(0, E)$, is $I \cdot \frac{\sqrt{2}}{\sqrt{\pi}}$-Lipschitz, where I is the Lipschitz constant of $J(x, y, \theta)$.*

Proof. Augmenting f with Gaussian noise, drawn from $\mathcal{N}(0, E)$, results in the so-called Weierstrass transform of f [33]:

$$\begin{aligned}\hat{f}(x) &= (f * \mathcal{N}(0, E))(x)\\ &= \frac{1}{(2\pi)^{n/2}} \int_{R^n} f(t) \exp\left(-\frac{1}{2}\|x-t\|^2\right) dt\\ &= \int_{R^n} f(t) \frac{1}{(2\pi)^{n/2}} \exp\left(-\frac{1}{2}\|x-t\|^2\right) dt\\ &= \int_{R^n} \nabla_x (J(t, y, \theta)) \frac{1}{(2\pi)^{n/2}} \exp\left(-\frac{1}{2}\|x-t\|^2\right) dt\end{aligned}$$

where t is obtained after adding noise to x.

It suffices to show $u \cdot \nabla \hat{f}(x) \leq I \cdot \sqrt{\frac{2}{\pi}}$. holds for any unit vector u. Note that:

$$\nabla \hat{f}(x) = \frac{1}{(2\pi)^{n/2}} \int_{R^n} f(t)(x-t) \exp\left(-\frac{1}{2}\|x-t\|^2\right) dt.$$

Thus, using $|f(t)| \leq I$, we get:

$$\begin{aligned}u \cdot \nabla \hat{f}(x) &\leq \frac{1}{(2\pi)^{n/2}} \int_{R^n} |I \cdot u \cdot (x-t)| \exp\left(-\frac{1}{2}\|x-t\|^2\right) dt\\ &= \frac{I}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} |s| \exp\left(-\frac{1}{2}s^2\right) ds = I \cdot \sqrt{\frac{2}{\pi}}.\end{aligned}$$

□

This theorem shows that augmenting noise sampled from $\mathcal{N}(0, E)$ bounds the gradients' maximum rate of change, making them smooth. The next theorem generalizes the result, showing that augmenting noise sampled from an isotropic Gaussian distribution, $\mathcal{N}(0, \sigma E)$, leads to smoother gradients the larger is σ. Said differently, when the distribution is less peaky, the Lipschitz constant of the $\hat{f}$, the augmented variant of the derivative, f, becomes smaller.

Theorem 4.2. *$\hat{f}$, attained by augmenting $\nabla_x J(x, y, \theta)$ with noise drawn from isotropic Gaussian distribution, $\mathcal{N}(0, \sigma E)$, is $I \cdot \frac{\sqrt{2}}{\sqrt{\pi}\sigma^2}$-Lipschitz, where I is the Lipschitz constant of $J(x, y, \theta)$.*

Proof. It suffices to show $u \cdot \nabla \hat{f}(x) \leq I \cdot \frac{\sqrt{2}}{\sqrt{\pi} * \sigma^2}$ holds for any unit vector u. Note that:

$$\begin{aligned}\hat{f}(x) &= \left(f * \mathcal{N}\left(0, \sigma^2\right)\right)(x)\\ &= \frac{1}{(2\pi)^{n/2} * \sigma} \int_{R^n} f(t) \exp\left(-\frac{1}{2}\|x-t\|^2/\sigma^2\right) dt\\ &= \int_{R^n} f(t) \frac{1}{(2\pi)^{n/2}/\sigma} \exp\left(-\frac{1}{2}\|x-t\|^2/\sigma^2\right) dt\\ &= \int_{R^n} \nabla_x (J(t, y, \theta)) \frac{1}{(2\pi)^{n/2} * \sigma} \exp\left(-\frac{1}{2}\|x-t\|^2/\sigma^2\right) dt.\end{aligned}$$

Thus, if we compute the derivative of $\hat{f}(x)$, we would get:

$$\nabla \hat{f}(x) = \frac{1}{(2\pi)^{n/2} * \sigma^3} \int_{R^n} f(t)(x-t) \exp\left(-\frac{1}{2}\|x-t\|^2/\sigma^2\right) dt$$

Now, using $|f(t)| \leq I$, we get:

$$\begin{aligned} u \cdot \nabla \hat{f}(x) &\leq \frac{1}{(2\pi)^{n/2} * \sigma^3} \int_{R^n} |I \cdot u \cdot (x-t)| \exp\left(-\frac{1}{2}\|x-t\|^2/\sigma^2\right) dt \\ &= \frac{I}{\sqrt{2\pi} * \sigma^3} \int_{-\infty}^{+\infty} |s| \exp\left(-\frac{1}{2}s^2/\sigma^2\right) ds \\ &= \frac{2I}{\sqrt{2\pi} * \sigma^3} \int_{0}^{+\infty} s \exp\left(-\frac{1}{2}s^2/\sigma^2\right) ds \\ &= \frac{2I}{\sqrt{2\pi} * \sigma^3} \int_{0}^{+\infty} \frac{1}{2} \exp\left(-\frac{1}{2}s^2/\sigma^2\right) ds^2 \\ &= \frac{2I}{\sqrt{2\pi} * \sigma^3} \int_{0}^{+\infty} \frac{1}{2} \exp\left(-\frac{1}{2}x/\sigma^2\right) dx \quad \left(\text{where } x = s^2\right) \\ &= \frac{2I}{\sqrt{2\pi} * \sigma^3} * \left(-\sigma * \exp\left(-\frac{1}{2}x/\sigma^2\right)\right)\Big|_0^{\infty} = I \cdot \frac{\sqrt{2}}{\sqrt{\pi} * \sigma^2}. \end{aligned}$$

□

This theorem demonstrates that, when augmenting noise sampled from an isotropic Gaussian distribution, we can smoothen the derivative, decreasing gradients' rate of change, by increasing σ. In return, the gradient estimates would become more consistent across attack iterations. This, we expect, renders the surrogate's peculiarities less likely to impact AEs, increasing the likelihood of generalization to the target.

Additive Noise from Smooth Distributions We extend the theoretical results, showing that we can smoothen f if we augment inputs with additive noise sampled from a smooth distribution $g(\cdot)$. Intuitively, a distribution $g(\cdot)$ is smooth if it does not exhibit sharp changes in its density within small regions. Formally, for a constant A, we define a distribution g as smooth if $|\int_{R^n} \nabla_x g(t-x)dt| \leq A$, where t is obtained by adding noise sampled from g to x. For example, for an isotropic Gaussian distribution, it can be shown that $A = \frac{\sqrt{2}}{\sqrt{\pi\sigma^2}}$. Given this definition, we generalize the previous theoretical results, and show that the Lipschitz constant of $\hat{f}$, obtained by adding noise sampled from g to f, is bounded.

Theorem 4.3. *$\hat{f}$, attained by augmenting $\nabla_x J(x, y, \theta)$ with noise drawn from a distribution g, such that $|\int_{R^n} \nabla_x g(t-x)dt| \leq A$, is $I \cdot A$-Lipschitz, where I is the Lipschitz constant of $J(x, y, \theta)$.*

Proof. Again, it suffices to show $u \cdot \nabla \hat{f}(x) \leq I \cdot A$. holds for any unit-norm vector u. Note that:

$$\hat{f}(x) = f * g(x) = \int_{R^n} f(t)g(t-x)dt$$

Thus:

$$\begin{aligned} u \cdot \nabla \hat{f}(x) &= \int_{R^n} f(t) \cdot u \cdot \nabla_t g(t-x)dt \\ \leq |\int_{R^n} f(t) \cdot u \cdot \nabla_t g(t-x)dt| &\leq \int_{R^n} |f(t) \cdot u| \cdot |\nabla_t g(t-x)|dt \\ &\leq \int_{R^n} I \cdot |\nabla_t g(t-x)|dt \leq I|\int_{R^n} \nabla_x g(t-x)dt| = I \cdot A \end{aligned}$$

□

This theorem demonstrates that there exist data augmentations other than Gaussian noise that can enable us to obtain a smooth $\hat{f}$. While we cannot ascertain that all augmentations we use satisfy the pre-conditions necessary for the theorem to hold (i.e., smooth distribution g), we find that, in practice, the augmentations we use lead to smoother gradient estimates (§6.7). In return, we also find that these augmentations help boost transferability.

5 Experimental Setup

Now we present the experimental setup. Our code is publicly available [56].

Data We used an ImageNet-compatible dataset [13] and CIFAR-10 [22] for evaluation, per common practice (e.g., [8, 53]). The former contains 1K 224×224 dimensional images pertaining to 1K classes, originally collected for the NeurIPS 2017 adversarial ML competition. For the latter, we sampled 1K 32×32 dimensional test-set images, roughly balanced between the dataset's ten classes. As the findings across datasets are consistent, we present detailed results primarily on ImageNet.

Models We used 18 DNNs to transfer attacks from (as surrogates) and to (as targets)—six for CIFAR-10 and 12 for ImageNet. To facilitate comparison with prior work, we included models widely used for testing transferability (e.g., [46, 53]). Furthermore, to ensure our findings are general, we included models covering varied architectures, including Inception, ResNet, VGG, DenseNet, MobileNet, ViT, and NASNet. Of the 12 ImageNet models, eight were normally trained and four were adversarially trained. Specifically, for normally trained models, we selected: Inc-v3 [39]; Inc-v4; IncRes-v2 [37]); Res-50; Res-101; Res-152 [15]; MNAS [41]; and ViT [9]. For adversarially trained models, we used: Inc-v3$_{adv}$ [23]; Inc-v3$_{ens3}$; Inc-v3$_{ens4}$; and IncRes-v2$_{ens}$ [43]. All six CIFAR-10 DNNs were normally trained. For this dataset, we used pre-trained VGG [36]; Res [15]; DenseNet [17]; MobileNet [34]; GoogleNet [38]; and Inc [39]. We obtained the models' PyTorch implementations and weights from public GitHub repositories [31, 32, 54].

Attacks We tested standard attack configurations and validated findings with varied perturbation norms, similar to prior work [46, 53]. Namely, we evaluated untargeted MI-FGSM-based attacks, bounded in ℓ_∞-norm. For ImageNet, unless stated otherwise, we tested $\epsilon = \frac{16}{255}$, but also experimented with $\epsilon \in \{\frac{8}{255}, \frac{24}{255}\}$ (App. D). For CIFAR-10, we experimented with $\epsilon \in \{0.02, 0.04\}$. We quantified attack success via *transferability rates*—the percentages of attempts at which AEs created against surrogates were misclassified by targets. As baselines, we used four state-of-the-art transferability-based attacks: Admix-DT, DST, VMI-DST, and UNDP-DT (see §3.1). App. B reports the parameters used in attacks and augmentation methods. Besides the four state-of-the-art baselines we tested, we considered including other recent attacks in the evaluation [18, 50, 58]. However, these either lacked publicly available implementations [50, 58], or achieved uncompetitive transferability rates in our experiments [18].

6 Experimental Results

We start by evaluating individual augmentation methods and standard combinations with scaling, diverse inputs, and translations (§6.1). We then turn to analyzing *all possible* compositions between a representative subset of augmentation types via exhaustive search

to assess whether transferability typically improves when considering more augmentations (§6.2). Next, we present the results of genetic search (§6.3). In §6.4, we report evaluate UltCombBase—the best combination found by exhaustive search—and UltCombGen—the best combination found by genetic search—including on defended models, and provide comparisons with the baselines attacks. We then compare parallel and serial composition, showing the former improves transferability (§6.5), before turning to attack runtimes (§6.6). Last, we provide empirical support to the theoretical results presented in §4 (§6.7).

6.1 Individual Augmentations

Initially, we evaluated transferability integrating a single augmentation at a time in attacks, or when composing individual augmentations with leading augmentations proposed in prior work [25, 26, 46]—namely, diverse inputs, scaling, and translation (DST). To this end, we selected ten augmentations to represent augmentations categories (Tab. 1). We found that considering each of the ten augmentations individually does *not* lead to competitive performance with the baselines. However, composing individual augmentations with DST enhanced transferability markedly. Tab. 1 shows transferability rates on ImageNet. Surprisingly, simple augmentations in color-space fared particularly well, outperforming most baselines and all advanced augmentation methods (e.g., AutoAugment) in most cases. Particularly, Composing GS with DST (GS-DST attack) performed best in this setting, outperforming all baselines but VMI-DST. GS-DST attained strong results when target models were either normally or adversarially trained (Tabs. 2–3), with different ϵs, and on CIFAR-10.

6.2 Exhaustive Search

We wanted to evaluate whether transferability is monotonic in the number of augmentation types—i.e., whether composing more techniques increases or, at least, preserves transferability. To this end, we ran exhaustive search (§3.4) over augmentations covering all categories and assessed the effect of composing more augmentations on transferability. Note, however, that running exhaustive search with the 46 augmentations we consider as well as DST and Admix (i.e., a total of 48 augmentation methods) would be prohibitive. Thus, we selected the best performing augmentation method of each of the seven categories (Tab. 1) and DST, and evaluated all 2^7 (=128) possible compositions (per §3.3). Specifically, we tested every possible combination of GS, CutOut, Sharp, NeuTrans, AutoAugment, Admix, and DST. Given a composition, we produced AEs against the Inc-v3 ImageNet DNN, and computed the expected transferability rate against remaining ImageNet DNNs. Then, for every pair of attacks differing only in whether a single augmentation method was incorporated in the composition, we tested whether adding the augmentation method improved transferability.

The results showed a mostly monotonic relationship between transferability and augmentations. Except for NeuTrans and Sharp, which sometimes harmed transferability, composing more augmentation methods increased or preserved transferability. I.e., with a few exceptions, augmentation compositions containing a superset of augmentations compared to other compositions typically had larger or equal transferability. Notably, comparing all compositions enabled us to find that a composition of all seven augmentation methods except for NeuTrans attained the best transferability. We call this composition UltCombBase.

6.3 Genetic Search

While we considered a more comprehensive set of augmentations than prior work to construct UltCombBase, this set was still relatively restricted. To discover combinations that advance transferability further while taking all augmentations into account, we ran genetic search (§3.4). Initially, to set the genetic search hyperparameters and find how close to optimal are the combinations found by genetic search, we ran it on the set of seven augmentations tested with exhaustive search. Specifically, we varied the number of generations (n_{gen}) and population sizes in genetic search and measured *(1)* the fraction of the search space that would be covered by genetic search compared to exhaustive search; and *(2)* the ratio between the average transferability rates attained by the fittest combination found by genetic search and the optimal combination included in UltCombBase. Our aim was to find whether genetic search is capable of discovering augmentation combinations that achieve nearly as high transferability as exhaustive search (i.e., maximizing the second metric) while searching a small fraction of the search space (i.e., minimizing the second metric). We repeated the experiment 100 times, using Inc-v3 as a surrogate and the remaining ImageNet models as targets. Because we tested all augmentation combinations in exhaustive search, we did not need to reproduce AEs when running genetic search. After hyperparameter sweep, we set p_{cross}=60%, p_{mutate}=10%, and p_{aug}=55% for best performance.

The results showed that with as litts as n_{gen}=2 and population size of 20, it is possible to find compositions attaining >99% of UltCombBase's transferability rates. Accordingly, using these hyperparameters, we ran genetic search on all 46 augmentations we considered for the first time to boost transferability, as well as DST and Admix, for a total of 48 augmentation techniques. Again, we selected Inc-v3 as a surrogate and the remaining ImageNet models as targets. The search process found a combination of 33 augmentations that achieved the highest transferability (see App. C for the complete list). We refer to the resulting attack composing all 33 combinations by UltCombGen.

6.4 The Most Effective Combinations

We evaluated the transferability of AEs produced by composing the augmentations found via exhaustive and genetic search (UltCombBase and UltCombGen, respectively) extensively, testing transferability to normally and adversarially trained DNNs, various defenses, and commercial systems. Additionally, we also explored the role of the number of augmented images (sample size) and augmentation methods for boosting transferability.

Normally and Adversarially Trained Targets The UltCombBase obtained higher transferability to normally trained models than the baselines (89.6% vs. ≤85.0% avg. transferability; Tab. 2). This held across different values of ϵ (Tab. 8 in App. D), and on the CIFAR-10 dataset with different architectures. UltCombGen, composing more augmentations, reached even higher transferability rates, with 92.6% average transferability rate to normally trained models on ImageNet.

Table 1: Avg. transferability (%) on ImageNet from Inc-v3 to all other models when integrating individual augmentations composed with DST. A vertical line separates the baselines. from our attacks.

Attack	Admix-DT	DST	VMI-DST	UNDP-DT	CS-DST	CJ-DST	fPCA-DST
Avg.	80.6	81.8	**88.8**	79.8	84.8	85.9	83.5
Attack (cont.)	GS-DST	RE-DST	CutMix-DST	CutOut-DST	NeuTrans-DST	Sharp-DST	AutoAugment-DST
Avg. (cont.)	*87.0*	84.9	54.0	84.1	73.5	80.1	82.9

Table 2: Average transferability (%) of black-box attacks on ImageNet, from all surrogates to normally and adversarially trained targets. A vertical line separates the baselines from our attacks.

Targets	**Admix-DT**	**DST**	**VMI-DST**	**UNDP-DT**	**GS-DST**	**UltComb$_{\text{Base}}$**	**UltComb$_{\text{Gen}}$**
Normally trained	79.7	82.1	82.9	85.0	86.2	*89.6*	**92.6**
Adversarially trained	78.2	77.4	82.5	65.4	83.1	*85.3*	**91.8**

Table 3: Transferability (%) on ImageNet, from an ensemble of normally trained surrogates (incl. Inc-v4, Res-50, Res-101 and Res-152) to adversarially trained targets. A vertical line separates the baselines from our attacks.

Model	**Admix-DT**	**DST**	**VMI-DST**	**UNDP-DT**	**GS-DST**	**UltComb$_{\text{Base}}$**	**UltComb$_{\text{Gen}}$**
Inc-v3$_{adv}$	89.3	89.1	92.8	88.2	92.6	*93.2*	**96.2**
Inc-v3$_{ens3}$	90.0	89.6	93.8	90.6	93.5	*95.4*	**96.3**
Inc-v3$_{ens4}$	89.0	87.8	93.1	86.1	92.3	*93.4*	**96.4**
IncRes-v2$_{ens}$	84.8	83.4	90.1	75.5	88.6	*91.1*	**94.6**

Furthermore, UltComb$_{\text{Base}}$ and UltComb$_{\text{Gen}}$ achieved the highest performance also when transferring attacks to adversarially trained DNNs (Tab. 2; 85.3% and 91.8% avg. transferability for UltComb$_{\text{Base}}$ and UltComb$_{\text{Gen}}$, respectively, compared to ≤82.5% avg. transferability for the baselines). Transferring AEs crafted using an ensemble of models increased transferability further (Tab. 3; respectively, 93.4% and 95.7% avg. transferability for UltComb$_{\text{Base}}$ and UltComb$_{\text{Gen}}$). Per a paired t-test, the differences between UltComb$_{\text{Base}}$ and UltComb$_{\text{Gen}}$ compared to the baselines over all pairs of surrogates and targets considered were statistically significant (p-value<0.01).

Additional Defenses Besides adversarially trained models, we evaluated transferability against five standard defenses. Two defenses, bit reduction (Bit-Red) [52] and neural representation purification (NRP) [29], seek to sanitize adversarial perturbations. Two others, randomized smoothing (RS) [4] and randomized smoothing with adversarial training (ARS) [33] offer provable robustness guarantees. Last, TRS leverages an ensemble of smooth DNNs trained to have misaligned gradients, to defend attacks [53]. We evaluated all defenses but TRS on ImageNet. We used the defenses with default parameters, and transferred AEs crafted against an ensemble of normally trained models. The results are shown in Tab. 4. Except for NRP, where VMI-DST attained the highest performance, UltComb$_{\text{Gen}}$ outperformed the baselines by large margins. Following [53], we tested TRS on CIFAR-10 with adversarial perturbation norms $\epsilon \in \{0.02, 0.04\}$. UltComb's variants did best against this defense as well (Tab. 5).

Commercial System Last, we tested attacks against Google Cloud Vision to simulate an even more realistic setting where target models are not publicly available and have been trained on a distinct training set compared to the surrogate. To estimate transferability, we classified the benign ImageNet images via Google's API and calculated the ratio of AEs that were classified differently than their benign counterparts. Tab. 6 presents the result. UltComb$_{\text{Gen}}$ outperformed *all* baselines, with 0.8%–9.5% higher transferability rates.

6.5 Parallel vs. Serial Composition

To corroborate that parallel composition is useful for transferability, we compared it to the previously established serial composition of the augmentation methods. In this experiment, we tested the augmentations included in UltComb and measured transferability on ImageNet, with Inc-v3 as surrogate and all other models as targets. The results showed that serial composition led to markedly lower transferability than parallel composition—26.1% vs. 88.9% average transferability rates. A potential explanation is that serial composition deteriorates image quality, significantly dropping the surrogate's benign accuracy (e.g., 21.0% vs. 82.4% benign accuracy on Inc-v3 on augmented images when using serial and parallel composition, respectively), such that the adversarial directions produced do not generalize to the target models.

6.6 Attack Run-Times

MI-FGSM's time complexity is predominated by the gradient computation steps. Accordingly, the attacks' run times are directly affected by the number of samples the augmentation methods create

Table 4: Transferability (%) from an ensemble of normally trained surrogates (Inc-v4, Res-50, Res-101 and Res-152) to defended ImageNet models.

Defense	Admix-DT	DST	VMI-DST	UNDP-DT	UltComb$_{\text{Base}}$	UltComb$_{\text{Gen}}$
Bit-Red	88.6	88.2	94.8	94.9	**96.0**	*95.5*
NRP	51.0	54.9	**80.0**	27.9	*65.3*	55.8
RS	87.3	84.8	90.6	85.5	*88.5*	**95.6**
ARS	65.4	62.9	66.5	61.9	*67.0*	**71.9**

Table 5: Transferability (%) on CIFAR-10 from a normally trained VGG surrogate to an ensemble of Res DNNs trained via TRS.

Epsilon	Admix-DT	DST	VMI-DST	UNDP-DT	GS-DST	UltComb$_{\text{Base}}$	UltComb$_{\text{Gen}}$
0.02	23.7	26.6	22.1	24.9	28.2	*32.7*	**46.5**
0.04	45.8	46.7	41.9	52.6	52.8	*59.4*	**80.0**

Table 6: Transferability (%) on ImageNet from an esemble of normally trained surrogates (Inc-v4, Res-50, Res-101 and Res-152) to Google Cloud Vision.

Admix-DT	DST	VMI-DST	UNDP-DT	UltComb$_{\text{Base}}$	UltComb$_{\text{Gen}}$
76.2	73.4	72.6	*81.3*	76.5	**82.1**

(i.e., samples emitted by $D(\cdot)$ in Alg. 1): The more samples emitted by the augmentation method, the more time would be spent computing gradients for updating the AEs in each iteration, thus increasing AEs' generation time. The empirical measurements corroborate this intuition (Tab. 7). DST augments MI-FGSM with the least samples (except for UNDP-DT that runs for 100 iterations) and runs for 10 iterations, leading to the fastest attack (DST). GS-DST is the second fastest attack, while UltComb$_{\text{Base}}$ is slower than Admix-DT but substantially faster (×2.44) than VMI-DST. UltComb$_{\text{Gen}}$ augments attacks with the largest number of samples, thus increasing run time the most. We note that no particular effort was invested to make GS-DST, UltComb$_{\text{Base}}$, and UltComb$_{\text{Gen}}$ more time-efficient (e.g., stacking augmented samples for parallel computation, similarly to Admix-DT). Moreover, since transferability-based attacks generate AEs offline, and only once per surrogate model, attack run-time is a marginal consideration for selecting an attack compared to transferability. Lastly, for improved time efficiency, an adversary with strict time requirements may consider using a subset of samples augmented by UltComb$_{\text{Gen}}$'s. We found that when sampling 30 of the 165 augmented images (UltComb$_{\text{Gen}}$-30 in Tab. 7), we obtain an attack ×7.27 faster than UltComb$_{\text{Gen}}$ with roughly the same transferability. By contrast, increasing the number of samples in baseline attacks only slows them down with barely no increase in transferability.

6.7 Supporting Theory

Our theoretical results show that augmentations often smoothen the gradients computed by the surrogate, thus decreasing the effect of the surrogate's peculiarities on AEs. In turn, this is likely to lead to improve the generalization of AEs from the surrogate to target models. Our empirical findings support this theory. Particularly, using the Inc-v3 models as a surrogate, we tested the (cosine) similarity between gradients throughout attack iterations when integrating different augmentations into attacks. Additionally, we

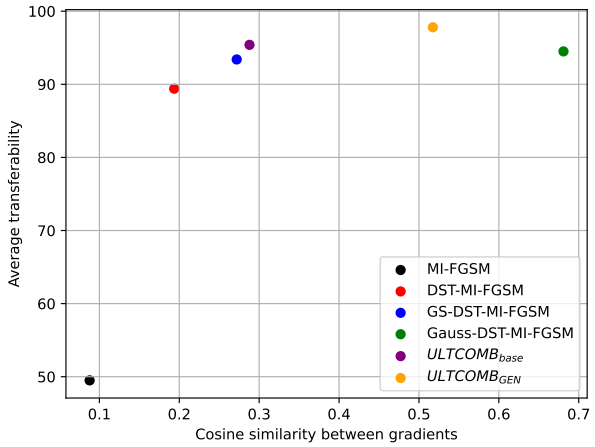

Figure 1: The relationship between the similarity between consecutive gradients computed in attacks and transferability rates. Results obtained using Inc-v3 as a surrogate and the remaining ImageNet models as targets. Gauss-DST refers attacks composing Gaussian noise and DST and uses the same number of augmented images as UltComb$_{\text{Gen}}$. Notice how UltComb$_{\text{Base}}$ and UltComb$_{\text{Gen}}$ lead to higher similarity between gradients than most attacks, and how the transferability rates tend to increase as the gradient similarity increases.

Table 7: The number of samples augmented and the avg. time of crafting an AE per attack. Times were measured on an Nvidia A5000 GPU, on ImageNet, when attacking an Inc-v3, and averaged for 1K samples. For best transferability rates, UNDP-DT was run for 100 iterations, while other attacks were run for 10.

Attack	Admix-DT	DST	VMI-DST	UNDP-DT	GS-DST	UltComb$_{\text{Base}}$	UltComb$_{\text{Gen}}$-30	UltComb$_{\text{Gen}}$
Augmented samples	15	5	105	1	10	30	30	165
Time (s)	1.68	**0.72**	11.29	1.65	*1.10*	3.62	5.54	40.27

measured the avg. transferability of attacks to the remaining ImageNet models to assess the relationship between smoothness and transferability. We expected that Gaussian-noise augmentations as well as compositions of multiple augmentations (as in UltComb$_{\text{Base}}$ and UltComb$_{\text{Gen}}$) would lead to smoother gradients, with higher cosine similarities across attack iterations. Moreover, we anticipated that transferability would be higher when the gradients are smoother. As shown in Fig. 1, both of these expectations held—the cosine similarities between consecutive gradients were higher for UltComb$_{\text{Base}}$ and UltComb$_{\text{Gen}}$ than for all other attacks except for when Gaussian noise is integrated, and the transferability rates tended to be higher as the cosine similarities increased. We note that the method integrating Gaussian noise attains the highest cosine similarities with slightly lower transferability than UltComb$_{\text{Base}}$. Accordingly, in addition to smoothness, there may be other factors that can influence transferability. We leave the identification of these factors for future work.

7 Conclusion

Leveraging a newly proposed means to integrate data augmentations into attacks (i.e., parallel composition) and systematically studying a broad range of augmentation methods, our work uncovered a mostly monotonic relationship between augmentations and transferability. Our approach helped us identify compositions (UltComb$_{\text{Base}}$ and UltComb$_{\text{Gen}}$) that outperform prior techniques integrated into attacks; these should be considered as a standard baseline in follow-up work on transferability. Our work also puts forward empirically supported theoretical explanations for why augmentations help boost transferability.

Acknowledgments

This work has been supported in part by grant No. 2023641 from the United States-Israel Binational Science Foundation (BSF); by a grant from the Blavatnik Interdisciplinary Cyber Research Center (ICRC); by Intel® via a Rising Star Faculty Award; by the Israel Science Foundation (grant number 1807/23); by a gift from KDDI Research; by Len Blavatnik and the Blavatnik Family foundation; by a Maof prize for outstanding young scientists; by the Ministry of Innovation, Science & Technology, Israel (grant number 0603870071); by NVIDIA via a hardware grant; and by a grant from the Tel Aviv University Center for AI and Data Science (TAD).

References

[1] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *ECML*.
[2] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2018. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*.
[3] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE S&P*.
[4] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *ICML*.
[5] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. 2019. AutoAugment: Learning augmentation policies from data. In *CVPR*.
[6] Terrance DeVries and Graham W Taylor. 2017. Improved regularization of convolutional neural networks with Cutout. *arXiv* (2017).
[7] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In *CVPR*.
[8] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. 2019. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *CVPR*.
[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.
[10] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *CVPR*.
[11] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2015. A neural algorithm of artistic style. *arXiv* (2015).
[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *ICLR*.
[13] Google Brain. 2017. NeurIPS 2017: Adversarial Learning Development Set. https://bit.ly/3fq4pN6.
[14] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. 2018. Countering adversarial images using input transformations. In *ICLR*.
[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *ECCV*.
[16] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2021. Meta-learning in neural networks: A survey. *IEEE PAMI* 44, 9 (2021), 5149–5169.
[17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *CVPR*.
[18] Qian Huang, Isay Katsman, Horace He, Zeqi Gu, Serge Belongie, and Ser-Nam Lim. 2019. Enhancing adversarial example transferability with an intermediate level attack. In *ICCV*.
[19] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. 2019. Prior convictions: Black-box adversarial attacks with bandits and priors. In *ICLR*.
[20] Alexander Jung and Vicram Rajagopalan. 2020. imgaug. https://imgaug.readthedocs.io. Accessed on 2024-04-19.
[21] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications* 80 (2021), 8091–8126.
[22] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report. University of Toronto.
[23] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial machine learning at scale. In *ICLR*.
[24] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *ICLRW*.
[25] Qizhang Li, Yiwen Guo, Wangmeng Zuo, and Hao Chen. 2023. Towards Evaluating Transfer-based Attacks Systematically, Practically, and Fairly. In *NeurIPS*.
[26] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. 2020. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *ICLR*.
[27] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into transferable adversarial examples and black-box attacks. In *ICLR*.
[28] Yuhao Mao, Chong Fu, Saizhuo Wang, Shouling Ji, Xuhong Zhang, Zhenguang Liu, Jun Zhou, Alex X Liu, Raheem Beyah, and Ting Wang. 2022. Transfer attacks revisited: A large-scale empirical study in real computer vision settings. In *IEEE S&P*.
[29] Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. 2020. A self-supervised approach for adversarial robustness. In *CVPR*.
[30] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial

settings. In *Euro S&P*.
[31] Huy Phan, David Widmann, Zafar, and Heon Song. [n. d.]. PyTorch models trained on CIFAR-10. https://github.com/huyvnphan/PyTorch_CIFAR10.
[32] PyTorch Core Team. [n. d.]. Torchvision. https://pytorch.org/vision.
[33] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. 2019. Provably robust deep learning via adversarially trained smoothed classifiers. In *NeurIPS*.
[34] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*.
[35] Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of big data* 6, 1 (2019), 1–48.
[36] K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.
[37] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *AAAI*.
[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR*.
[39] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.
[40] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *ICLR*.
[41] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. 2019. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *CVPR*.
[42] Florian Tramer. 2022. Detecting adversarial examples is (nearly) as hard as classifying them. In *ICML*.
[43] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble adversarial training: Attacks and defenses. In *ICLR*.
[44] Aladin Virmaux and Kevin Scaman. 2018. Lipschitz regularity of deep neural networks: Analysis and efficient estimation. In *NeurIPS*.
[45] Xiaosen Wang and Kun He. 2021. Enhancing the transferability of adversarial attacks through variance tuning. In *CVPR*.
[46] Xiaosen Wang, Xuanran He, Jingdong Wang, and Kun He. 2021. Admix: Enhancing the transferability of adversarial attacks. In *ICCV*.
[47] Xiaosen Wang, Jiadong Lin, Han Hu, Jingdong Wang, and Kun He. 2021. Boosting adversarial transferability through enhanced momentum. In *BMVC*.
[48] Xin Wang, Jie Ren, Shuyun Lin, Xiangming Zhu, Yisen Wang, and Quanshi Zhang. 2020. A Unified Approach to Interpreting and Boosting Adversarial Transferability. In *ICLR*.
[49] Ren Wu, Shengen Yan, Yi Shan, Qingqing Dang, and Gang Sun. 2015. Deep image: Scaling up image recognition. *arXiv* (2015).
[50] Weibin Wu, Yuxin Su, Michael R Lyu, and Irwin King. 2021. Improving the transferability of adversarial samples with adversarial transformations. In *CVPR*.
[51] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. 2019. Improving transferability of adversarial examples with input diversity. In *CVPR*.
[52] Weilin Xu, David Evans, and Yanjun Qi. 2018. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *NDSS*.
[53] Zhuolin Yang, Linyi Li, Xiaojun Xu, Shiliang Zuo, Qian Chen, Pan Zhou, Benjamin Rubinstein, Ce Zhang, and Bo Li. 2021. TRS: Transferability reduced ensemble via promoting gradient diversity and model smoothness. In *NeurIPS*.
[54] Shengming Yuan and Qilong Zhang. [n. d.]. TF to PyTorch Model. https://github.com/ylhz/tf_to_pytorch_model.
[55] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. CutMix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*.
[56] Zebin Yun, Achi-Or Weingarten, Eyal Ronen, and Mahmood Sharif. 2024. Implementation of UltComb. https://github.com/yundaqwe/Ultimate-Combo.
[57] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *ICLR*.
[58] Jianping Zhang, Jen-tse Huang, Wenxuan Wang, Yichen Li, Weibin Wu, Xiaosen Wang, Yuxin Su, and Michael R Lyu. 2023. Improving the Transferability of Adversarial Samples by Path-Augmented Method. In *CVPR*.
[59] Yechao Zhang, Shengshan Hu, Leo Yu Zhang, Junyu Shi, Minghui Li, Xiaogeng Liu, Wei Wan, and Hai Jin. 2024. Why Does Little Robustness Help? Understanding and Improving Adversarial Transferability from Surrogate Training. In *S&P*.
[60] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2020. Random erasing data augmentation. In *AAAI*.

A Other Augmentations Studied

Besides the 12 augmentations in §3.2, we considered the following 34 augmentations.

Color-space: *(1) K-means color quantization*; *(2) Jpeg compression*; *(3) Voronoi*; *(4) Random invert*; *(5) Random posterize*; *(6) Random solarize*; *(7) Random equalize*; *(8) Superpixels*; *(9) Guassian noise*; *(10) Emboss*; *(11) fancy PCA (fPCA).*

Random Deletion: *(1) Dropout*; *(2) Random crop.*

Kernel Filters: *(1) Gaussian blur*; *(2) Average blur*; *(3) Median blur*; *(4) Bilateral blur*; *(5) Motion blur*; *(6) MeanShift blur*; *(7) Edge detect*; *(8) Canny*; *(9) Average pooling*; *(10) Max pooling*; *(11) Min Poling*; *(12) Median pooling.*

Style Transfer: *(1) Clouds*; *(2) Fog*; *(3) Frost*; *(4) Snow*; *(5) Rain.*

Spatial: *(1) Random perspective*; *(2 Elastic transform*; *(3) Random vertical flip*; *(4) Patch shuffle.*

B Attack and Augmentation Method Parameters

Similarly to standard practice [47], we set the MI-FGSM decay factor μ=1.0 , and attacks' number of iterations T=10, and their step-size $\alpha = \frac{\epsilon}{T}$. The only exception is UNDP-DT found to attain low transferability rates with these parameters. Therefore, for UNDP-DT, we use default parameters found to achieve best performance by Li et al. [25]: T=100 and $\alpha = \frac{1}{255}$.

We mostly used default or commonly used parameters of augmentation methods. For CJ, we performed random adjustments of image hue $\in [-0.5, 0.5]$, contrast $\in [0.5, 1.5]$, saturation $\in [0.5, 1.5]$, and brightness$\in [0.5, 1.5]$. For CutOut, we replaced values in selected regions with zeros, and the portion of masked areas compared to image dimensions lied in $[0.02, 0.4]$, with aspect ratios $\in [0.4, 2.5]$. In comparison, for RE, the dimension of masked areas relatively to the image dimensions lied in $[0.02, 0.2]$, with aspect ratios $\in [0.3, 3.3]$. For Sharp, we used the following edge-enhancement mask:

$$\begin{bmatrix} -0.5 & -0.5 & -0.5 \\ -0.5 & 5.0 & -0.5 \\ -0.5 & -0.5 & -0.5 \end{bmatrix}.$$

For diverse inputs, images were transformed with probability 0.5. For the Admix operation, consistently with [46], we randomly sampled three images from other categories for mixing as part of the Admix-DT attack. However, for the interest of computational efficiency, we use only one image for mixing when composing Admix with other augmentation methods. We did not find that mixing with fewer images harmed performance. In fact, it even improved transferability in some cases. Finally, in CutMix, we picked the top left coordinate (r_x, r_y), the width, r_w, and height, r_h, of the region to be replaced, using the formulas:

$$r_x \sim \mathcal{U}(0, W), \quad r_w = W\sqrt{1-\lambda},$$

$$r_y \sim \mathcal{U}(0, H), \quad r_h = H\sqrt{1-\lambda},$$

where $\mathcal{U}$ is the uniform distribution, W is the image width, H is the image height, and λ is a parameter set to 0.5.

In an attempt to enhance transferability further, we optimized the parameters of a few augmentation methods we considered via

Table 8: Transferability rates (%) on ImageNet, from a Inc-v3 surrogate to other normally trained models, with perturbation norms $\epsilon \in \{\frac{8}{255}, \frac{24}{255}\}$ other than the default $\epsilon = \frac{16}{255}$.

ϵ	Attack	Inc-v3	Inc-v4	Res-152	IncRes-v2	Res-50	Res-101	ViT	MNAS	Inc-v3$_{adv}$	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$	Avg.
8/255	Admix-DT	98.7	75.0	60.6	69.3	68.3	62.7	33.5	67.8	57.6	52.5	51.8	36.1	57.7
	DST	**99.7**	77.1	62.9	70.9	69.8	64.8	34.1	71.1	61.4	55.0	54.7	35.9	59.8
	VMI-DST	99.5	80.0	68.1	75.1	74.8	68.9	39.7	74.5	68.6	64.4	64.5	47.7	66.0
	UNDP-DT	99.4	**92.2**	**82.4**	**90.6**	**85.6**	**85.1**	32.3	*86.6*	48.7	43.9	40.8	20.3	64.4
	GS-DST	99.5	81.3	71.1	77.6	77.3	72.0	37.0	75.4	69.2	63.9	63.9	45.2	66.7
	UltComb$_{Base}$	**99.7**	86.0	75.0	81.8	81.2	76.3	40.8	82.5	72.7	67.2	64.2	46.2	*70.4*
	UltComb$_{Gen}$	**99.7**	*89.8*	*80.7*	*87.1*	*84.3*	*81.5*	**52.6**	**87.8**	**80.7**	**77.3**	**75.4**	**58.4**	**77.8**
24/255	Admix-DT	99.9	97.1	93.7	95.9	94.8	94.4	73.4	94.3	87.1	88.6	88.2	79.9	89.8
	DST	**100.0**	97.8	93.6	96.8	95.1	93.9	74.9	95.9	84.6	90.5	89.3	77.2	90.0
	VMI-DST	**100.0**	97.9	95.0	97.1	96.2	95.6	78.4	97.1	89.3	93.2	91.9	84.0	92.3
	UNDP-DT	99.8	98.9	97.3	98.9	97.9	98.0	78.8	*97.6*	84.0	82.9	78.4	59.8	88.4
	GS-DST	**100.0**	98.6	95.7	98.3	96.7	97.0	86.7	97.5	94.5	95.9	94.9	90.4	95.1
	UltComb$_{Base}$	**100.0**	*99.3*	*97.4*	*99.2*	*97.5*	*98.4*	*83.8*	*99.0*	*92.5*	*95.1*	*95.0*	*86.9*	*94.9*
	UltComb$_{Gen}$	**100.0**	**100.0**	**99.8**	**100.0**	**99.6**	**99.9**	**94.1**	**99.5**	**97.9**	**99.3**	**98.4**	**95.2**	**98.5**

grid search. Except for the Gaussian kernel's size used in translation-invariant attacks [8], we found that the selected parameters had little impact on transferability. Specifically, for translations, after considering Gaussian kernels of sizes $\in \{5 \times 5, 7 \times 7, 9 \times 9\}$, we set the default to 7×7, except for Admix-DT, for which the 9×9 kernel performed best. The results show that our choice of Admix parameters (m=1 and Gaussian kernel size of 9×9) improves its performance. For GS, we found ω_R, ω_G, and ω_B had little impact on transferability, as long as the weight assigned to each channel was >0.1. Accordingly, we set ω_R, ω_G, and ω_B to 0.299, 0.587, and 0.114, respectively, per commonly used values (e.g., in the Python `PyTorch` package[1]). Finally, for CS, we only swapped the blue and green channels, as this led to a minor improvement compared to swapping all three channels.

Finally, we clarify that each of our attack combinations emits the original image once, alongside the transformed images. Moreover, when aggregating the gradients, the gradients of the original and transformed images are assigned equal weights. We tested whether weighting the gradients differently (e.g., assigning higher or lower weight to the original sample) can help improve transferability using the GS method. However, we found that equal weights attained the best results.

C Augmentations Included in UltComb$_{Gen}$

The following augmentations are composed together in UltComb$_{Gen}$: DST; CutMix; random crop; random rotation; CJ; Gaussian blur; random affine; random perspective; fPCA; elastic transform; horizontal flip; Admix; random invert; random solarize; AutoAugment; NeuTrans; CutOut; JPEG compression; K-means color quantization; superpixels; average blur; median blur; motion blur; MeanShift blur; edge detect; canny; average pooling; min pooling; patch shuffle; fog; frost; rain; Guassian noise.

D Transferability Rates on ImageNet

Tab. 8 shows the transferability rates on ImageNet from Inc-v3 to other normally traiend models with varied perturbation norms (i.e., values of ϵ).

[1]https://bit.ly/3ynCyUD