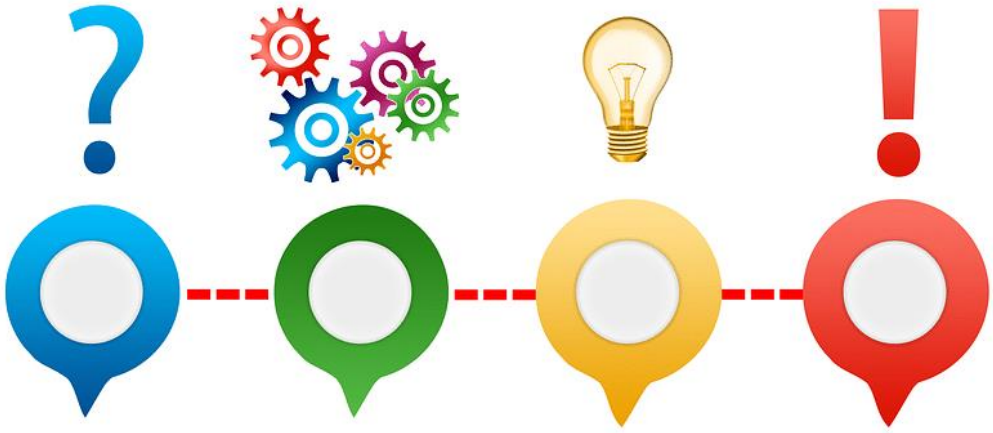
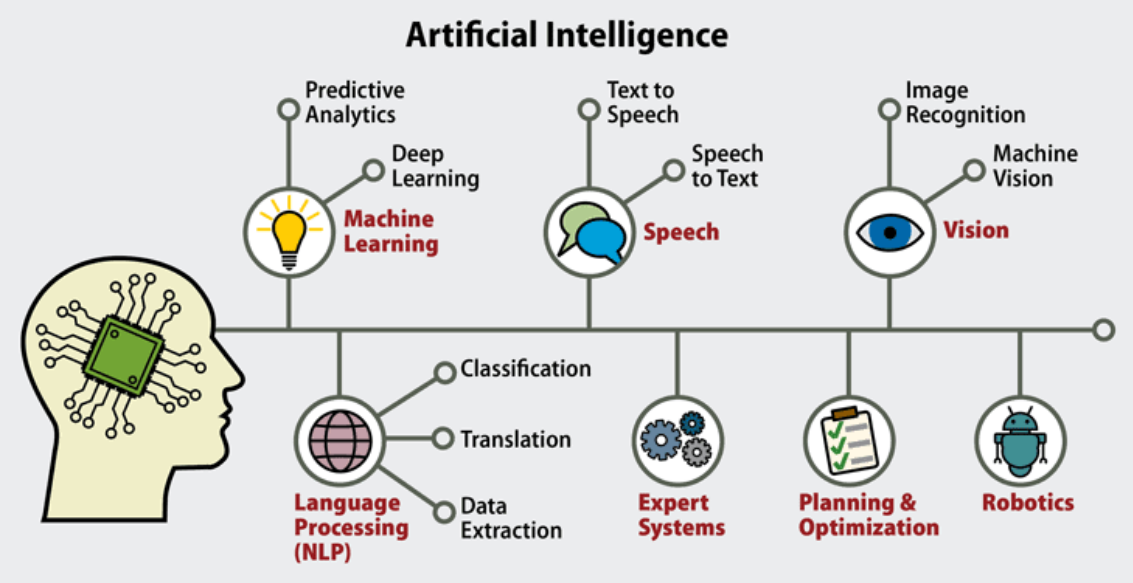
10 Cutting Edge Research-Papers In Computer Vision and Image Generation



These papers provide a breakneck pace breadth of information about computer vision and image generation that is generally useful in the interdisciplinary scientific field that deals with how computers can be made to gain high-level understanding from digital images or videos.



Contents

- Spherical CNNs
- Adversarial Examples that Fool both Computer Vision and Time-Limited Humans
- Group Normalization
- A Closed-form Solution to Photorealistic Image Stylization
- Taskonomy: Disentangling Task Transfer Learning
- GANimation: Anatomically-aware Facial Animation from a Single Image
- Self-Attention Generative Adversarial Networks
- Video-to-Video Synthesis
- Everybody Dance Now
- Large Scale GAN Training for High Fidelity Natural Image Synthesis

SPHERICAL CNNS

Taco S. Cohen*
University of Amsterdam

Mario Geiger*
EPFL

Jonas Köhler*
University of Amsterdam

Max Welling
University of Amsterdam & CIFAR

ABSTRACT

Convolutional Neural Networks (CNNs) have become the method of choice for learning problems involving 2D planar images. However, a number of problems of recent interest have created a demand for models that can analyze spherical images. Examples include omnidirectional vision for drones, robots, and autonomous cars, molecular regression problems, and global weather and climate modelling. A naive application of convolutional networks to a planar projection of the spherical signal is destined to fail, because the space-varying distortions introduced by such a projection will make translational weight sharing ineffective.

In this paper we introduce the building blocks for constructing spherical CNNs. We propose a definition for the spherical cross-correlation that is both expressive and rotation-equivariant. The spherical correlation satisfies a generalized Fourier theorem, which allows us to compute it efficiently using a generalized (non-commutative) Fast Fourier Transform (FFT) algorithm. We demonstrate the computational efficiency, numerical accuracy, and effectiveness of spherical CNNs applied to 3D model recognition and atomization energy regression.

1 INTRODUCTION

Convolutional networks are able to detect local patterns regardless of their position in the image. Like patterns in a planar image, patterns on the sphere can move around, but in this case the "move" is a 3D rotation instead of a translation. In analogy to the planar CNN, we would like to build a network that can detect patterns regardless of how they are rotated over the sphere.

As shown in Figure 1, there is no good way to use translational convolution or cross-correlation[1] to analyze spherical signals. The most obvious approach, then, is to change the definition of cross-correlation by replacing filter translations by rotations. Doing so, we run into a subtle but important difference between the plane and the sphere: whereas the space of moves for the plane (2D translations) is itself isomorphic to the plane, the space of moves for the sphere (3D rotations) is a different, *three-dimensional* manifold called $\mathrm{SO}(3)$[2]. It follows that the result of a spherical correlation (the output feature map) is to be considered a signal on $\mathrm{SO}(3)$, not a signal on the sphere, S^2. For this reason, we deploy $\mathrm{SO}(3)$ group correlation in the higher layers of a spherical CNN (Cohen and Welling, 2016).

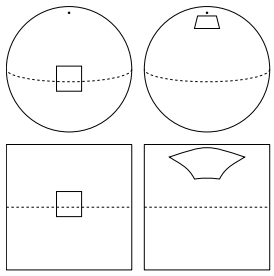


Figure 1: Any planar projection of a spherical signal will result in distortions. Rotation of a spherical signal cannot be emulated by translation of its planar projection.

*Equal contribution.

[1]Despite the name, CNNs typically use cross-correlation instead of convolution in the forward pass. In this paper we will generally use the term cross-correlation, or correlation for short.

[2]To be more precise: although the symmetry group of the plane contains more than just translations, the translations form a subgroup that acts on the plane. In the case of the sphere there is no coherent way to define a composition for points on the sphere, and so the sphere cannot act on itself (it is not a group). For this reason, we must consider the whole of $\mathrm{SO}(3)$.

The implementation of a spherical CNN (S^2-CNN) involves two major challenges. Whereas a square grid of pixels has discrete translation symmetries, no perfectly symmetrical grids for the sphere exist. This means that there is no simple way to define the rotation of a spherical filter by one pixel. Instead, in order to rotate a filter we would need to perform some kind of interpolation. The other challenge is computational efficiency; $\mathrm{SO}(3)$ is a three-dimensional manifold, so a naive implementation of $\mathrm{SO}(3)$ correlation is $O(n^6)$.

We address both of these problems using techniques from non-commutative harmonic analysis (Chirikjian and Kyatkin, 2001; Folland, 1995). This field presents us with a far-reaching generalization of the Fourier transform, which is applicable to signals on the sphere as well as the rotation group. It is known that the $\mathrm{SO}(3)$ correlation satisfies a Fourier theorem with respect to the $\mathrm{SO}(3)$ Fourier transform, and the same is true for our definition of S^2 correlation. Hence, the S^2 and $\mathrm{SO}(3)$ correlation can be implemented efficiently using generalized FFT algorithms.

Because we are the first to use cross-correlation on a continuous group inside a multi-layer neural network, we rigorously evaluate the degree to which the mathematical properties predicted by the continuous theory hold in practice for our discretized implementation.

Furthermore, we demonstrate the utility of spherical CNNs for rotation invariant classification and regression problems by experiments on three datasets. First, we show that spherical CNNs are much better at rotation invariant classification of Spherical MNIST images than planar CNNs. Second, we use the CNN for classifying 3D shapes. In a third experiment we use the model for molecular energy regression, an important problem in computational chemistry.

CONTRIBUTIONS

The main contributions of this work are the following:

1. The theory of spherical CNNs.
2. The first automatically differentiable implementation of the generalized Fourier transform for S^2 and $\mathrm{SO}(3)$. Our PyTorch code is easy to use, fast, and memory efficient.
3. The first empirical support for the utility of spherical CNNs for rotation-invariant learning problems.

2 RELATED WORK

It is well understood that the power of CNNs stems in large part from their ability to exploit (translational) symmetries though a combination of weight sharing and translation equivariance. It thus becomes natural to consider generalizations that exploit larger groups of symmetries, and indeed this has been the subject of several recent papers by Gens and Domingos (2014); Olah (2014); Dieleman et al. (2015; 2016); Cohen and Welling (2016); Ravanbakhsh et al. (2017); Zaheer et al. (2017b); Guttenberg et al. (2016); Cohen and Welling (2017). With the exception of $\mathrm{SO}(2)$-steerable networks (Worrall et al., 2017; Weiler et al., 2017), these networks are all limited to discrete groups, such as discrete rotations acting on planar images or permutations acting on point clouds. Other very recent work is concerned with the analysis of spherical images, but does not define an equivariant architecture (Su and Grauman, 2017; Boomsma and Frellsen, 2017). Our work is the first to achieve equivariance to a continuous, non-commutative group ($\mathrm{SO}(3)$), and the first to use the generalized Fourier transform for fast group correlation. A preliminary version of this work appeared as Cohen et al. (2017).

To efficiently perform cross-correlations on the sphere and rotation group, we use generalized FFT algorithms. Generalized Fourier analysis, sometimes called abstract- or noncommutative harmonic analysis, has a long history in mathematics and many books have been written on the subject (Sugiura, 1990; Taylor, 1986; Folland, 1995). For a good engineering-oriented treatment which covers generalized FFT algorithms, see (Chirikjian and Kyatkin, 2001). Other important works include (Driscoll and Healy, 1994; Healy et al., 2003; Potts et al., 1998; Kunis and Potts, 2003; Drake et al., 2008; Maslen, 1998; Rockmore, 2004; Kostelec and Rockmore, 2007; 2008; Potts et al., 2009; Makadia et al., 2007; Gutman et al., 2008).

3 CORRELATION ON THE SPHERE AND ROTATION GROUP

We will explain the S^2 and $\mathrm{SO}(3)$ correlation by analogy to the classical planar $\mathbb{Z}^2$ correlation. The planar correlation can be understood as follows:

> The value of the output feature map at translation $x \in \mathbb{Z}^2$ is computed as an inner product between the input feature map and a filter, shifted by x.

Similarly, the spherical correlation can be understood as follows:

> The value of the output feature map evaluated at rotation $R \in \mathrm{SO}(3)$ is computed as an inner product between the input feature map and a filter, rotated by R.

Because the output feature map is indexed by a rotation, it is modelled as a function on $\mathrm{SO}(3)$. We will discuss this issue in more detail shortly.

The above definition refers to various concepts that we have not yet defined mathematically. In what follows, we will go through the required concepts one by one and provide a precise definition. Our goal for this section is only to present a mathematical model of spherical CNNs. Generalized Fourier theory and implementation details will be treated later.

The Unit Sphere S^2 can be defined as the set of points $x \in \mathbb{R}^3$ with norm 1. It is a two-dimensional manifold, which can be parameterized by spherical coordinates $\alpha \in [0, 2\pi]$ and $\beta \in [0, \pi]$.

Spherical Signals We model spherical images and filters as continuous functions $f : S^2 \to \mathbb{R}^K$, where K is the number of channels.

Rotations The set of rotations in three dimensions is called $\mathrm{SO}(3)$, the "special orthogonal group". Rotations can be represented by 3×3 matrices that preserve distance (i.e. $||Rx|| = ||x||$) and orientation ($\det(R) = +1$). If we represent points on the sphere as 3D unit vectors x, we can perform a rotation using the matrix-vector product Rx. The rotation group $\mathrm{SO}(3)$ is a three-dimensional manifold, and can be parameterized by ZYZ-Euler angles $\alpha \in [0, 2\pi]$, $\beta \in [0, \pi]$, and $\gamma \in [0, 2\pi]$.

Rotation of Spherical Signals In order to define the spherical correlation, we need to know not only how to rotate points $x \in S^2$ but also how to rotate filters (i.e. functions) on the sphere. To this end, we introduce the rotation operator L_R that takes a function f and produces a rotated function $L_R f$ by composing f with the rotation R^{-1}:

$$[L_R f](x) = f(R^{-1}x). \tag{1}$$

Due to the inverse on R, we have $L_{RR'} = L_R L_{R'}$.

Inner products The inner product on the vector space of spherical signals is defined as:

$$\langle \psi, f \rangle = \int_{S^2} \sum_{k=1}^{K} \psi_k(x) f_k(x) dx, \tag{2}$$

The integration measure dx denotes the standard rotation invariant integration measure on the sphere, which can be expressed as $d\alpha \sin(\beta) d\beta / 4\pi$ in spherical coordinates (see Appendix A). The invariance of the measure ensures that $\int_{S^2} f(Rx) dx = \int_{S^2} f(x) dx$, for any rotation $R \in \mathrm{SO}(3)$. That is, the volume under a spherical heightmap does not change when rotated. Using this fact, we can show that $L_{R^{-1}}$ is adjoint to L_R, which implies that L_R is unitary:

$$\begin{aligned} \langle L_R \psi, f \rangle &= \int_{S^2} \sum_{k=1}^{K} \psi_k(R^{-1}x) f_k(x) dx \\ &= \int_{S^2} \sum_{k=1}^{K} \psi_k(x) f_k(Rx) dx \\ &= \langle \psi, L_{R^{-1}} f \rangle. \end{aligned} \tag{3}$$

Spherical Correlation With these ingredients in place, we are now ready to state mathematically what was stated in words before. For spherical signals f and ψ, we define the correlation as:

$$[\psi \star f](R) = \langle L_R \psi, f \rangle = \int_{S^2} \sum_{k=1}^{K} \psi_k(R^{-1}x) f_k(x) dx. \tag{4}$$

As mentioned before, the output of the spherical correlation is a function on $\mathrm{SO}(3)$. This is perhaps somewhat counterintuitive, and indeed the conventional definition of spherical convolution gives as output a function on the sphere. However, as shown in Appendix B, the conventional definition effectively restricts the filter to be circularly symmetric about the Z axis, which would greatly limit the expressive capacity of the network.

Rotation of $\mathrm{SO}(3)$ Signals We defined the rotation operator L_R for spherical signals (eq. 1), and used it to define spherical cross-correlation (eq. 4). To define the $\mathrm{SO}(3)$ correlation, we need to generalize the rotation operator so that it can act on signals defined on $\mathrm{SO}(3)$. As we will show, naively reusing eq. 1 is the way to go. That is, for $f : \mathrm{SO}(3) \to \mathbb{R}^K$, and $R, Q \in \mathrm{SO}(3)$:

$$[L_R f](Q) = f(R^{-1}Q). \tag{5}$$

Note that while the argument $R^{-1}x$ in Eq. 1 denotes the rotation of $x \in S^2$ by $R^{-1} \in \mathrm{SO}(3)$, the analogous term $R^{-1}Q$ in Eq. 5 denotes to the composition of rotations (i.e. matrix multiplication).

Rotation Group Correlation Using the same analogy as before, we can define the correlation of two signals on the rotation group, $f, \psi : \mathrm{SO}(3) \to \mathbb{R}^K$, as follows:

$$[\psi \star f](R) = \langle L_R \psi, f \rangle = \int_{\mathrm{SO}(3)} \sum_{k=1}^{K} \psi_k(R^{-1}Q) f_k(Q) dQ. \tag{6}$$

The integration measure dQ is the invariant measure on $\mathrm{SO}(3)$, which may be expressed in ZYZ-Euler angles as $d\alpha \sin(\beta) d\beta d\gamma / (8\pi^2)$ (see Appendix A).

Equivariance As we have seen, correlation is defined in terms of the rotation operator L_R. This operator acts naturally on the input space of the network, but what justification do we have for using it in the second layer and beyond?

The justification is provided by an important property, shared by all kinds of convolution and correlation, called equivariance. A layer Φ is equivariant if $\Phi \circ L_R = T_R \circ \Phi$, for some operator T_R. Using the definition of correlation and the unitarity of L_R, showing equivariance is a one liner:

$$[\psi \star [L_Q f]](R) = \langle L_R \psi, L_Q f \rangle = \langle L_{Q^{-1}R} \psi, f \rangle = [\psi \star f](Q^{-1}R) = [L_Q [\psi \star f]](R). \tag{7}$$

The derivation is valid for spherical correlation as well as rotation group correlation.

4 Fast Spherical Correlation with G-FFT

It is well known that correlations and convolutions can be computed efficiently using the Fast Fourier Transform (FFT). This is a result of the Fourier theorem, which states that $\widehat{f * \psi} = \hat{f} \cdot \hat{\psi}$. Since the FFT can be computed in $O(n \log n)$ time and the product $\cdot$ has linear complexity, implementing the correlation using FFTs is asymptotically faster than the naive $O(n^2)$ spatial implementation.

For functions on the sphere and rotation group, there is an analogous transform, which we will refer to as the generalized Fourier transform (GFT) and a corresponding fast algorithm (GFFT). This transform finds it roots in the representation theory of groups, but due to space constraints we will not go into details here and instead refer the interested reader to Sugiura (1990) and Folland (1995).

Conceptually, the GFT is nothing more than the linear projection of a function onto a set of orthogonal basis functions called "matrix element of irreducible unitary representations". For the circle (S^1) or line ($\mathbb{R}$), these are the familiar complex exponentials $\exp(in\theta)$. For $\mathrm{SO}(3)$, we have the Wigner D-functions $D^l_{mn}(R)$ indexed by $l \geq 0$ and $-l \leq m, n \leq l$. For S^2, these are the spherical harmonics[3] $Y^l_m(x)$ indexed by $l \geq 0$ and $-l \leq m \leq l$.

Denoting the manifold (S^2 or $\mathrm{SO}(3)$) by X and the corresponding basis functions by U^l (which is either vector-valued (Y^l) or matrix-valued (D^l)), we can write the GFT of a function $f : X \to \mathbb{R}$ as

$$\hat{f}^l = \int_X f(x) \overline{U^l(x)} dx. \tag{8}$$

[3]Technically, S^2 is not a group and therefore does not have irreducible representations, but it is a quotient of groups $\mathrm{SO}(3)/\mathrm{SO}(2)$ and we have the relation $Y^l_m = D^l_{m0}|_{S^2}$

This integral can be computed efficiently using a GFFT algorithm (see Section 4.1).

The inverse $\mathrm{SO}(3)$ Fourier transform is defined as:

$$f(R) = \sum_{l=0}^{b} (2l+1) \sum_{m=-l}^{l} \sum_{n=-l}^{l} \hat{f}^l_{mn} D^l_{mn}(R), \tag{9}$$

and similarly for S^2. The maximum frequency b is known as the bandwidth, and is related to the resolution of the spatial grid (Kostelec and Rockmore, 2007).

Using the well-known (in fact, defining) property of the Wigner D-functions that $D^l(R)D^l(R') = D^l(RR')$ and $D^l(R^{-1}) = D^l(R)^\dagger$, it can be shown (see Appendix D) that the $\mathrm{SO}(3)$ correlation satisfies a Fourier theorem[4]: $\widehat{\psi \star f} = \hat{f} \cdot \hat{\psi}^\dagger$, where $\cdot$ denotes matrix multiplication of the two block matrices $\hat{f}$ and $\hat{\psi}^\dagger$.

Similarly, using $Y(Rx) = D(R)Y(x)$ and $Y^l_m = D^l_{m0}|_{S^2}$, one can derive an analogous S^2 convolution theorem: $\widehat{\psi \star f}^l = \hat{f}^l \cdot \hat{\psi}^{l\dagger}$, where $\hat{f}^l$ and $\hat{\psi}^l$ are now vectors. This says that the $\mathrm{SO}(3)$-FT of the S^2 correlation of two spherical signals can be computed by taking the outer product of the S^2-FTs of the signals. This is shown in figure 2.

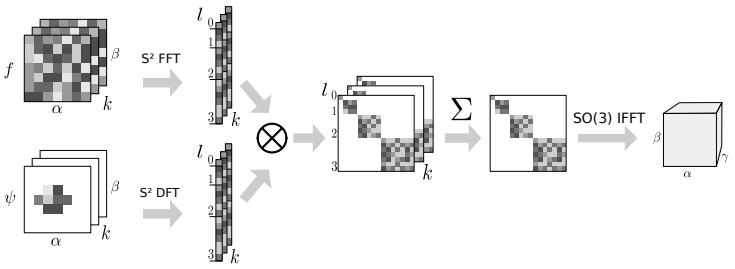


Figure 2: Spherical correlation in the spectrum. The signal f and the locally-supported filter ψ are Fourier transformed, block-wise tensored, summed over input channels, and finally inverse transformed. Note that because the filter is locally supported, it is faster to use a matrix multiplication (DFT) than an FFT algorithm for it. We parameterize the sphere using spherical coordinates α, β, and $\mathrm{SO}(3)$ with ZYZ-Euler angles α, β, γ.

4.1 Implementation of G-FFT and Spectral G-Conv

Here we sketch the implementation of GFFTs. For details, see (Kostelec and Rockmore, 2007).

The input of the $\mathrm{SO}(3)$ FFT is a spatial signal f on $\mathrm{SO}(3)$, sampled on a discrete grid and stored as a 3D array. The axes correspond to the ZYZ-Euler angles α, β, γ. The first step of the $\mathrm{SO}(3)$-FFT is to perform a standard 2D translational FFT over the α and γ axes. The FFT'ed axes correspond to the m, n axes of the result. The second and last step is a linear contraction of the β axis of the FFT'ed array with a precomputed array of samples from the Wigner-d (small-d) functions $d^l_{mn}(\beta)$. Because the shape of d^l depends on l (it is $(2l+1) \times (2l+1)$), this linear contraction is implemented as a custom GPU kernel. The output is a set of Fourier coefficients $\hat{f}^l_{mn}$ for $l \geq n, m \geq -l$ and $l = 0, \ldots, L_{\max}$.

The algorithm for the S^2-FFTs is very similar, only in this case we FFT over the α axis only, and do a linear contraction with precomputed Legendre functions over the β axis.

Our code is available at `https://github.com/jonas-koehler/s2cnn`.

5 Experiments

In a first sequence of experiments, we evaluate the numerical stability and accuracy of our algorithm. In a second sequence of experiments, we showcase that the new cross-correlation layers we have

[4]This result is valid for real functions. For complex functions, conjugate ψ on the left hand side.

introduced are indeed useful building blocks for several real problems involving spherical signals. Our examples for this are recognition of 3D shapes and predicting the atomization energy of molecules.

5.1 EQUIVARIANCE ERROR

In this paper we have presented the first instance of a group equivariant CNN for a continuous, non-commutative group. In the discrete case, one can prove that the network is exactly equivariant, but although we can prove $[L_R f] * \psi = L_R[f * \psi]$ for continuous functions f and ψ on the sphere or rotation group, this is not exactly true for the discretized version that we actually compute. Hence, it is reasonable to ask if there are any significant discretization artifacts and whether they affect the equivariance properties of the network. If equivariance can not be maintained for many layers, one may expect the weight sharing scheme to become much less effective.

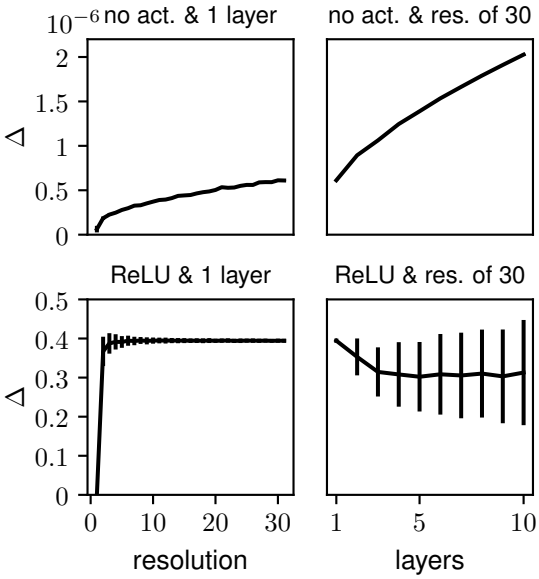


Figure 3: Δ as a function of the resolution and the number of layers.

We first tested the equivariance of the $\mathrm{SO}(3)$ correlation at various resolutions b. We do this by first sampling $n = 500$ random rotations R_i as well as n feature maps f_i with $K = 10$ channels. Then we compute $\Delta = \frac{1}{n}\sum_{i=1}^{n} \mathrm{std}(L_{R_i}\Phi(f_i) - \Phi(L_{R_i} f_i))/\mathrm{std}(\Phi(f_i))$, where Φ is a composition of $\mathrm{SO}(3)$ correlation layers with randomly initialized filters. In case of perfect equivariance, we expect this quantity to be zero. The results (figure 3 (top)), show that although the approximation error Δ grows with the resolution and the number of layers, it stays manageable for the range of resolutions of interest.

We repeat the experiment with ReLU activation function after each correlation operation. As shown in figure 3 (bottom), the error is higher but stays flat. This indicates that the error is not due to the network layers, but due to the feature map rotation, which is exact only for bandlimited functions.

5.2 ROTATED MNIST ON THE SPHERE

In this experiment we evaluate the generalization performance with respect to rotations of the input. For testing we propose a version MNIST dataset projected on the sphere (see fig. 4). We created two instances of this dataset: one in which each digit is projected on the northern hemisphere and one in which each projected digit is additionally randomly rotated.

Architecture and Hyperparameters As a baseline model, we use a simple CNN with layers conv-ReLU-conv-ReLU-FC-softmax, with filters of size 5×5, $k = 32, 64, 10$ channels, and stride 3 in both layers ($\approx$ 68K parameters). We compare to a spherical CNN with layers S^2conv-ReLU-$\mathrm{SO}(3)$conv-ReLU-FC-softmax, bandwidth $b = 30, 10, 6$ and $k = 20, 40, 10$ channels ($\approx$ 58K parameters).

Figure 4: Two MNIST digits projected onto the sphere using stereographic projection. Mapping back to the plane results in non-linear distortions.

Results We trained each model on the non-rotated (NR) and the rotated (R) training set and evaluated it on the non-rotated and rotated test set. See table 1. While the planar CNN achieves high accuracy in the NR / NR regime, its performance in the R / R regime is much worse, while the spherical CNN is unaffected. When trained on the

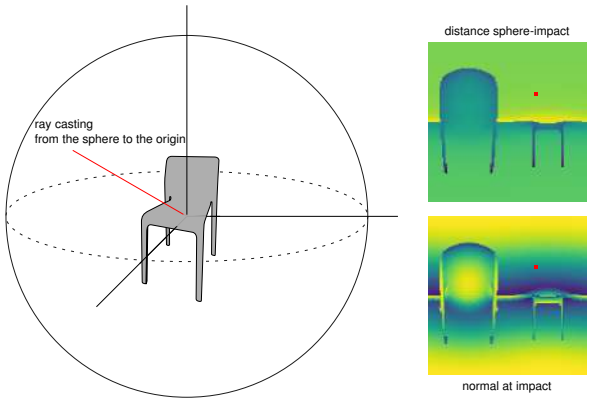


Figure 5: The ray is cast from the surface of the sphere towards the origin. The first intersection with the model gives the values of the signal. The two images of the right represent two spherical signals in (α, β) coordinates. They contain respectively the distance from the sphere and the cosine of the ray with the normal of the model. The red dot corresponds to the pixel set by the red line.

non-rotated dataset and evaluated on the rotated dataset (NR / R), the planar CNN does no better than random chance. The spherical CNN shows a slight decrease in performance compared to R/R, but still performs very well.

	NR / NR	R / R	NR / R
planar	0.98	0.23	0.11
spherical	0.96	0.95	0.94

Table 1: Test accuracy for the networks evaluated on the spherical MNIST dataset. Here R = rotated, NR = non-rotated and X / Y denotes, that the network was trained on X and evaluated on Y.

5.3 Recognition of 3D Shapes

Next, we applied S^2CNN to 3D shape classification. The SHREC17 task (Savva et al., 2017) contains 51300 3D models taken from the ShapeNet dataset (Chang et al., 2015) which have to be classified into 55 common categories (tables, airplanes, persons, etc.). There is a consistently aligned regular dataset and a version in which all models are randomly perturbed by rotations. We concentrate on the latter to test the quality of our rotation equivariant representations learned by S^2CNN.

Representation We project the 3D meshes onto an enclosing sphere using a straightforward ray casting scheme (see Fig. 5). For each point on the sphere we send a ray towards the origin and collect 3 types of information from the intersection: ray length and $\cos$ / $\sin$ of the surface angle. We further augment this information with ray casting information for the convex hull of the model, which in total gives us 6 channels for the signal. This signal is discretized using a Driscoll-Healy grid (Driscoll and Healy, 1994) with bandwidth $b = 128$. Ignoring non-convexity of surfaces we assume this projection captures enough information of the shape to be useful for the recognition task.

Architecture and Hyperparameters Our network consists of an initial S^2conv-BN-ReLU block followed by two $\mathrm{SO}(3)$conv-BN-ReLU blocks. The resulting filters are pooled using a max pooling layer followed by a last batch normalization and then fed into a linear layer for the final classification. It is important to note that the the max pooling happens over the group SO(3): if f_k is the k-th filter in the final layer (a function on SO(3)) the result of the pooling is $\max_{x \in SO(3)} f_k(x)$. We used 50, 70, and 350 features for the S^2 and the two SO(3) layers, respectively. Further, in each layer we reduce the resolution b, from 128, 32, 22 to 7 in the final layer. Each filter kernel ψ on SO(3) has non-local support, where $\psi(\alpha, \beta, \gamma) \neq 0$ iff $\beta = \frac{\pi}{2}$ and $\gamma = 0$ and the number of points of the discretization is proportional to the bandwidth in each layer. The final network contains ≈ 1.4M parameters, takes 8GB of memory at batch size 16, and takes 50 hours to train.

Method	P@N	R@N	F1@N	mAP	NDCG
Tatsuma_ReVGG	0.705	0.769	0.719	0.696	0.783
Furuya_DLAN	0.814	0.683	0.706	0.656	0.754
SHREC16-Bai_GIFT	0.678	0.667	0.661	0.607	0.735
Deng_CM-VGG5-6DB	0.412	0.706	0.472	0.524	0.624
Ours	0.701 (3rd)	0.711 (2nd)	0.699 (3rd)	0.676 (2nd)	0.756 (2nd)

Table 2: Results and best competing methods for the SHREC17 competition.

Results We evaluated our trained model using the official metrics and compared to the top three competitors in each category (see table 2 for results). Except for precision and F1@N, in which our model ranks third, it is the runner up on each other metric. The main competitors, Tatsuma_ReVGG and Furuya_DLAN use input representations and network architectures that are highly specialized to the SHREC17 task. Given the rather task agnostic architecture of our model and the lossy input representation we use, we interpret our models performance as strong empirical support for the effectiveness of Spherical CNNs.

5.4 Prediction of Atomization Energies from Molecular Geometry

Finally, we apply S^2CNN on molecular energy regression. In the QM7 task (Blum and Reymond, 2009; Rupp et al., 2012) the atomization energy of molecules has to be predicted from geometry and charges. Molecules contain up to $N = 23$ atoms of $T = 5$ types (H, C, N, O, S). They are given as a list of positions p_i and charges z_i for each atom i.

Representation by Coulomb matrices Rupp et al. (2012) propose a rotation and translation invariant representation of molecules by defining the *Coulomb matrix* $C \in \mathbb{R}^{N\times N}$ (CM). For each pair of atoms $i \neq j$ they set $C_{ij} = (z_i z_j)/(|p_i - p_j|)$ and $C_{ii} = 0.5 z_i^{2.4}$. Diagonal elements encode the atomic energy by nuclear charge, while other elements encode Coulomb repulsion between atoms. This representation is not permutation invariant. To this end Rupp et al. (2012) propose a distance measure between Coulomb matrices used within Gaussian kernels whereas Montavon et al. (2012) propose sorting C or random sampling index permutations.

Representation as a spherical signal We utilize spherical symmetries in the geometry by defining a sphere S_i around around p_i for each atom i. The radius is kept uniform across atoms and molecules and chosen minimal such that no intersections among spheres in the training set happen. Generalizing the Coulomb matrix approach we define for each possible z and for each point x on S_i potential functions $U_z(x) = \sum_{j\neq i, z_j = z} \frac{z_i \cdot z}{|x-p_i|}$ producing a T channel spherical signal for each atom in the molecule (see figure 6). This representation is invariant with respect to translations and equivariant with respect to rotations. However, it is still not permutation invariant. The signal is discretized using a Driscoll-Healy (Driscoll and Healy, 1994) grid with bandwidth $b = 10$ representing the molecule as a sparse $N \times T \times 2b \times 2b$ tensor.

Architecture and Hyperparameters We use a deep ResNet style S^2CNN. Each ResNet block is made of S^2/SO(3)conv-BN-ReLU-SO(3)conv-BN after which the input is added to the result. We share weights among atoms making filters permutation invariant, by pushing the atom dimension into the batch dimension. In each layer we downsample the bandwidth, while increasing the number of features F. After integrating the signal over SO(3) each molecule becomes a $N \times F$ tensor. For permutation invariance over atoms we follow Zaheer et al. (2017a) and embed each resulting feature vector of an atom into a latent space using a MLP ϕ. Then we sum these latent representations over the atom dimension and get our final regression value for the molecule by mapping with another MLP ψ. Both ϕ and ψ are jointly optimized. Training a simple MLP only on the 5 frequencies of atom types in a molecule already gives a RMSE of ~ 19. Thus, we train the S^2CNN on the residual only, which improved convergence speed and stability over direct training. The final architecture is sketched in table 3. It has about 1.4M parameters, consumes 7GB of memory at batch size 20, and takes 3 hours to train.

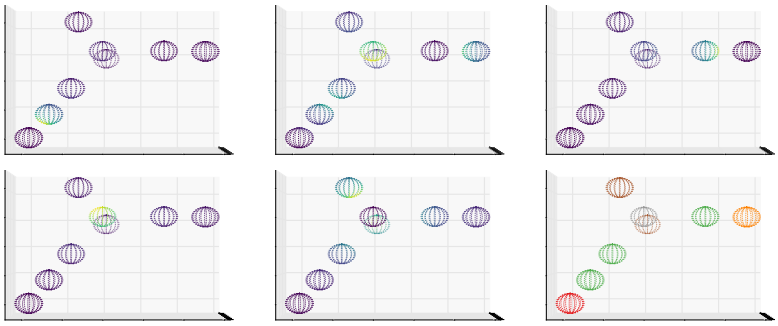


Figure 6: The five potential channels U_z with $z \in \{1, 6, 7, 8, 16\}$ for a molecule containing atoms H (red), C (green), N (orange), O (brown), S (gray).

Method	Author	RMSE
MLP / random CM	(a)	5.96
LGIKA(RF)	(b)	10.82
RBF kernels / random CM	(a)	11.40
RBF kernels / sorted CM	(a)	12.59
MLP / sorted CM	(a)	16.06
Ours		**8.47**

S^2CNN	Layer	Bandwidth	Features
	Input		5
	ResBlock	10	20
	ResBlock	8	40
	ResBlock	6	60
	ResBlock	4	80
	ResBlock	2	160
DeepSet	Layer	Input/Hidden	
	ϕ (MLP)	160/150	
	ψ (MLP)	100/50	

Table 3: Left: Experiment results for the QM7 task: (a) Montavon et al. (2012) (b) Raj et al. (2016). Right: ResNet architecture for the molecule task.

Results We evaluate by RMSE and compare our results to Montavon et al. (2012) and Raj et al. (2016) (see table 3). Our learned representation outperforms all kernel-based approaches and a MLP trained on sorted Coulomb matrices. Superior performance could only be achieved for an MLP trained on randomly permuted Coulomb matrices. However, sufficient sampling of random permutations grows exponentially with N, so this method is unlikely to scale to large molecules.

6 Discussion & Conclusion

In this paper we have presented the theory of Spherical CNNs and evaluated them on two important learning problems. We have defined S^2 and $\mathrm{SO}(3)$ cross-correlations, analyzed their properties, and implemented a Generalized FFT-based correlation algorithm. Our numerical results confirm the stability and accuracy of this algorithm, even for deep networks. Furthermore, we have shown that Spherical CNNs can effectively generalize across rotations, and achieve near state-of-the-art results on competitive 3D Model Recognition and Molecular Energy Regression challenges, without excessive feature engineering and task-tuning.

For intrinsically volumetric tasks like 3D model recognition, we believe that further improvements can be attained by generalizing further beyond $\mathrm{SO}(3)$ to the roto-translation group $\mathrm{SE}(3)$. The development of Spherical CNNs is an important first step in this direction. Another interesting generalization is the development of a Steerable CNN for the sphere (Cohen and Welling, 2017), which would make it possible to analyze vector fields such as global wind directions, as well as other sections of vector bundles over the sphere.

Perhaps the most exciting future application of the Spherical CNN is in omnidirectional vision. Although very little omnidirectional image data is currently available in public repositories, the increasing prevalence of omnidirectional sensors in drones, robots, and autonomous cars makes this a very compelling application of our work.

REFERENCES

L. C. Blum and J.-L. Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J. Am. Chem. Soc.*, 131:8732, 2009.

W. Boomsma and J. Frellsen. Spherical convolutions and their application in molecular modelling. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3436–3446. Curran Associates, Inc., 2017.

A.X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

G.S. Chirikjian and A.B. Kyatkin. *Engineering Applications of Noncommutative Harmonic Analysis*. CRC Press, 1 edition, may 2001. ISBN 9781420041767.

T.S. Cohen and M. Welling. Group equivariant convolutional networks. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, volume 48, pages 2990–2999, 2016.

T.S. Cohen and M. Welling. Steerable CNNs. In *ICLR*, 2017.

T.S. Cohen, M. Geiger, J. Koehler, and M. Welling. Convolutional networks for spherical signals. In *ICML Workshop on Principled Approaches to Deep Learning*, 2017.

S. Dieleman, K. W. Willett, and J. Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(2), 2015.

S. Dieleman, J. De Fauw, and K. Kavukcuoglu. Exploiting Cyclic Symmetry in Convolutional Neural Networks. In *International Conference on Machine Learning (ICML)*, 2016.

J.B. Drake, P.H. Worley, and E.F. D'Azevedo. Algorithm 888: Spherical harmonic transform algorithms. *ACM Trans. Math. Softw.*, 35(3):23:1–23:23, 2008. doi: 10.1145/1391989.1404581.

J.R. Driscoll and D.M. Healy. Computing Fourier transforms and convolutions on the 2-sphere. *Advances in applied mathematics*, 1994.

G.B. Folland. *A Course in Abstract Harmonic Analysis*. CRC Press, 1995.

R. Gens and P. Domingos. Deep Symmetry Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

B. Gutman, Y. Wang, T. Chan, P.M. Thompson, and others. Shape registration with spherical cross correlation. *2nd MICCAI workshop*, 2008.

N. Guttenberg, N. Virgo, O. Witkowski, H. Aoki, and R. Kanai. Permutation-equivariant neural networks applied to dynamics prediction. 2016.

D. Healy, D. Rockmore, P. Kostelec, and S. Moore. FFTs for the 2-Sphere – Improvements and Variations. *The journal of Fourier analysis and applications*, 9(4):340–385, 2003.

P.J. Kostelec and D.N. Rockmore. SOFT: SO(3) Fourier Transforms. 2007. URL `http://www.cs.dartmouth.edu/~geelong/soft/soft20_fx.pdf`.

P.J. Kostelec and D.N. Rockmore. FFTs on the rotation group. *Journal of Fourier Analysis and Applications*, 14(2):145–179, 2008.

S. Kunis and D. Potts. Fast spherical Fourier algorithms. *Journal of Computational and Applied Mathematics*, 161:75–98, 2003.

A. Makadia, C. Geyer, and K. Daniilidis. Correspondence-free structure from motion. *Int. J. Comput. Vis.*, 75(3):311–327, December 2007.

D.K. Maslen. Efficient Computation of Fourier Transforms on Compact Groups. *Journal of Fourier Analysis and Applications*, 4(1), 1998.

G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, O.A. von Lilienfeld, and K. Müller. Learning invariant representations of molecules for atomization energy prediction. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 449–457. 2012.

L. Nachbin. *The Haar Integral*. 1965.

C. Olah. Groups and Group Convolutions, 2014. URL `https://colah.github.io/posts/2014-12-Groups-Convolution/`.

D. Potts, G. Steidl, and M. Tasche. Fast and stable algorithms for discrete spherical Fourier transforms. *Linear Algebra and its Applications*, 275:433–450, 1998.

D. Potts, J. Prestin, and A. Vollrath. A fast algorithm for nonequispaced Fourier transforms on the rotation group. *Numerical Algorithms*, pages 1–28, 2009.

A. Raj, A. Kumar, Y. Mroueh, P.T. Fletcher, et al. Local group invariant representations via orbit embeddings. *arXiv preprint arXiv:1612.01988*, 2016.

S. Ravanbakhsh, J. Schneider, and B. Poczos. Deep learning with sets and point clouds. In *International Conference on Learning Representations (ICLR) – workshop track*, 2017.

D.N. Rockmore. Recent Progress and Applications in Group FFTS. *NATO Science Series II: Mathematics, Physics and Chemistry*, 136:227–254, 2004.

M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical Review Letters*, 108:058301, 2012.

M. Savva, F. Yu, H. Su, A. Kanezaki, T. Furuya, R. Ohbuchi, Z. Zhou, R. Yu, S. Bai, X. Bai, M. Aono, A. Tatsuma, S. Thermos, A. Axenopoulos, G. Th. Papadopoulos, P. Daras, X. Deng, Z. Lian, B. Li, H. Johan, Y. Lu, and S. Mk. Large-Scale 3D Shape Retrieval from ShapeNet Core55. In Ioannis Pratikakis, Florent Dupont, and Maks Ovsjanikov, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2017. ISBN 978-3-03868-030-7. doi: 10.2312/3dor.20171050.

Y.C. Su and K. Grauman. Learning spherical convolution for fast features from 360 imagery. *Adv. Neural Inf. Process. Syst.*, 2017.

M. Sugiura. *Unitary Representations and Harmonic Analysis*. John Wiley & Sons, New York, London, Sydney, Toronto, 2nd edition, 1990.

M.E. Taylor. *Noncommutative Harmonic Analysis*. American Mathematical Society, 1986. ISBN 0821815237.

M. Weiler, F.A. Hamprecht, and M. Storath. Learning steerable filters for rotation equivariant CNNs. 2017.

D.E. Worrall, S.J. Garbin, D. Turmukhambetov, and G.J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *CVPR*, 2017.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017a.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R.R. Salakhutdinov, and A.J. Smola. Deep sets. In *Advances in Neural Information Processing Systems 30*, pages 3393–3403, 2017b.

APPENDIX A: PARAMETERIZATION OF AND INTEGRATION ON S^2 AND $\mathrm{SO}(3)$

We use the ZYZ Euler parameterization for $\mathrm{SO}(3)$. An element $R \in \mathrm{SO}(3)$ is written as

$$R = R(\alpha, \beta, \gamma) = Z(\alpha)Y(\beta)Z(\gamma), \tag{10}$$

where $\alpha \in [0, 2\pi]$, $\beta \in [0, \pi]$ and $\gamma \in [0, 2\pi]$, and Z resp. Y are rotations around the Z and Y axes.

Using this parameterization, the normalized Haar measure is

$$dR = \frac{d\alpha}{2\pi} \frac{d\beta \sin(\beta)}{2} \frac{d\gamma}{2\pi} \tag{11}$$

We have $\int_{\mathrm{SO}(3)} dR = 1$. The Haar measure (Nachbin, 1965; Chirikjian and Kyatkin, 2001) is sometimes called the invariant measure because it has the property that $\int_{\mathrm{SO}(3)} f(R'R)dR = \int_{\mathrm{SO}(3)} f(R)dR$ (this is analogous to the more familiar property $\int_{\mathbb{R}} f(x+y)dx = \int_{\mathbb{R}} f(x)dx$ for functions on the line). This invariance property allows us to do many useful substitutions.

We have a related parameterization for the sphere. An element $x \in S^2$ is written

$$x(\alpha, \beta) = Z(\alpha)Y(\beta)n \tag{12}$$

where n is the north pole.

This parameterization makes explicit the fact that the sphere is a quotient $S^2 = \mathrm{SO}(3)/\mathrm{SO}(2)$, where $H = \mathrm{SO}(2)$ is the subgroup of rotations around the Z axis. Elements of this subgroup H leave the north pole invariant, and have the form $Z(\gamma)$. The point $x(\alpha, \beta) \in S^2$ is associated with the coset representative $\bar{x} = R(\alpha, \beta, 0) \in \mathrm{SO}(3)$. This element represents the coset $\bar{x}H = \{R(\alpha, \beta, \gamma) | \gamma \in [0, 2\pi]\}$.

The normalized Haar measure for the sphere is

$$dx = \frac{d\alpha}{2\pi} \frac{d\beta \sin\beta}{2} \tag{13}$$

The normalized Haar measure for $\mathrm{SO}(2)$ is

$$dh = \frac{d\gamma}{2\pi} \tag{14}$$

So we have $dR = dx\, dh$, again reflecting the quotient structure.

We can think of a function on S^2 as a γ-invariant function on $\mathrm{SO}(3)$. Given a function $f : S^2 \to \mathbb{C}$ we associate the function $\bar{f}(\alpha, \beta, \gamma) = f(\alpha, \beta)$. When using normalized Haar measures, we have:

$$\begin{aligned}
\int_{\mathrm{SO}(3)} \bar{f}(R)dR &= \frac{1}{8\pi^2} \int_0^{2\pi} d\alpha \int_0^{\pi} \sin\beta d\beta \int_0^{2\pi} d\gamma \bar{f}(\alpha, \beta, \gamma) \\
&= \frac{1}{8\pi^2} \int_0^{2\pi} d\alpha \int_0^{\pi} \sin\beta d\beta f(\alpha, \beta) \int_0^{2\pi} d\gamma \\
&= \frac{1}{4\pi} \int_0^{2\pi} d\alpha \int_0^{\pi} \sin\beta d\beta f(\alpha, \beta) \\
&= \int_{S^2} f(x)dx
\end{aligned} \tag{15}$$

This will allow us to define the Fourier transform on S^2 from the Fourier transform on $\mathrm{SO}(3)$, by viewing a function on S^2 as a γ-invariant function on $\mathrm{SO}(3)$ and taking its $\mathrm{SO}(3)$-Fourier transform.

APPENDIX B: CORRELATION & EQUIVARIANCE

We have defined the S^2 correlation as

$$[\psi \star f](R) = \langle L_R \psi, f \rangle = \int_{S^2} \sum_{k=1}^{K} \psi_k(R^{-1}x) f_k(x) dx. \tag{16}$$

Without loss of generality, we will analyze here the single-channel case $K = 1$.

This operation is equivariant:

$$\begin{aligned}
[\psi \star [L_Q f]](R) &= \int_{S^2} \psi(R^{-1}x) f(Q^{-1}x) dx \\
&= \int_{S^2} \psi(R^{-1}Qx) f(x) dx \\
&= \int_{S^2} \psi((Q^{-1}R)^{-1}x) f(x) dx \\
&= [\psi \star f](Q^{-1}R) \\
&= [L_Q[\psi \star f]](R)
\end{aligned} \tag{17}$$

A similar derivation can be made for the $\mathrm{SO}(3)$ correlation.

The spherical convolution defined by Driscoll and Healy (1994) is:

$$[f * \psi](x) = \int_{\mathrm{SO}(3)} f(Rn)\psi(R^{-1}x) dR \tag{18}$$

where n is the north pole. Note that in this definition, the output of the spherical convolution is a function on the sphere, not a function on $\mathrm{SO}(3)$ as in our definition of cross-correlation. Note further that unlike our definition, this definition involves an integral over $\mathrm{SO}(3)$.

If we write out the integral in terms of Euler angles, noting that the north-pole n is invariant to Z-axis rotations by γ, i.e. $R(\alpha, \beta, \gamma)n = Z(\alpha)Y(\beta)Z(\gamma)n = Z(\alpha)Y(\beta)n$, we see that this definition implicitly integrates over γ in only one of the factors (namely ψ), making it invariant wrt γ rotation. In other words, the filter is first "averaged" (making it circularly symmetric) before it is combined with f (This was observed before by Makadia et al. (2007)). We consider this to be much too limited for the purpose of pattern matching in spherical CNNs.

APPENDIX C: GENERALIZED FOURIER TRANSFORM

With each compact topological group (like $\mathrm{SO}(3)$) is associated a discrete set of orthogonal functions that arise as matrix elements of irreducible unitary representations of these groups. For the circle (the group $\mathrm{SO}(2)$) these are the complex exponentials (in the complex case) or sinusoids (for real functions). For $\mathrm{SO}(3)$, these functions are known as the Wigner D-functions.

As discussed in the paper, the Wigner D-functions are parameterized by a degree parameter $l \geq 0$ and order parameters $m, n \in [-l, \dots, l]$. In other words, we have a set of matrix-valued functions $D^l : \mathrm{SO}(3) \to \mathbb{C}^{(2l+1)\times(2l+1)}$.

The Wigner D-functions are orthogonal:

$$\langle D^l_{mn}, D^{l'}_{m'n'} \rangle = \int_0^{2\pi} \frac{d\alpha}{2\pi} \int_0^{\pi} \frac{d\beta \sin\beta}{2} \int_0^{2\pi} \frac{d\gamma}{2\pi} D^l_{mn}(\alpha, \beta, \gamma) \overline{D^{l'}_{m'n'}}(\alpha, \beta, \gamma) = \frac{\delta_{ll'}\delta_{mm'}\delta_{nn'}}{2l+1} \tag{19}$$

Furthermore, they are complete, meaning that any well behaved function $f : \mathrm{SO}(3) \to \mathbb{C}$ can be written as a linear combination of Wigner D-functions. This is the idea of the Generalized Fourier Transform $\mathcal{F}$ on $\mathrm{SO}(3)$:

$$f(R) = [\mathcal{F}^{-1}\hat{f}](R) = \sum_{l=0}^{\infty} (2l+1) \sum_{m=-l}^{l} \sum_{n=-l}^{l} \hat{f}^l_{mn} D^l_{mn}(R) \tag{20}$$

where $\hat{f}^l_{mn}$ are called the Fourier coefficients of f. Using the orthogonality property of the Wigner D-functions, one can see that the Fourier coefficients can be retrieved by computing the inner product with the Wigner D-functions:

$$\begin{aligned}
[\mathcal{F}f]^l_{mn} &= \int_{\mathrm{SO}(3)} f(R)\overline{D^l_{mn}(R)}dR \\
&= \int_{\mathrm{SO}(3)} \left[\sum_{l'=0}^{\infty}(2l'+1)\sum_{m'=-l'}^{l'}\sum_{n'=-l'}^{l'}\hat{f}^{l'}_{m'n'}D^{l'}_{m'n'}(R)\right]\overline{D^l_{mn}(R)}dR \\
&= \sum_{l'=0}^{\infty}(2l'+1)\sum_{m'=-l'}^{l'}\sum_{n'=-l'}^{l'}\hat{f}^{l'}_{m'n'}\int_{\mathrm{SO}(3)} D^{l'}_{m'n'}(R)\overline{D^l_{mn}}dR \\
&= \hat{f}^l_{mn}
\end{aligned} \tag{21}$$

APPENDIX D: FOURIER THEOREMS

Fourier convolution theorems for $\mathrm{SO}(3)$ and $\S^2$ can be found in Kostelec and Rockmore (2008); Makadia et al. (2007); Gutman et al. (2008). We derive them here for completeness.

To derive the convolution theorems, we will use the defining property of the Wigner D-matrices: that they are (irreducible, unitary) *representations* of $\mathrm{SO}(3)$. This means that they satisfy:

$$D^l(R)D^l(R') = D^l(RR'), \tag{22}$$

for any $R, R' \in \mathrm{SO}(3)$. Notice that the complex exponentials satisfy an analogous criterion for the circle group $S^1 \cong \mathrm{SO}(2)$. That is, $e^{inx}e^{iny} = e^{in(x+y)}$, where $x + y$ is the group operation for $\mathrm{SO}(2)$.

Unitarity means that $D^l(R)D^{l\dagger}(R) = I$. Irreducibility means, essentially, that the set of matrices $\{D^l(R) \,|\, R \in \mathrm{SO}(3)\}$ cannot be simultaneously block-diagonalized.

To derive the Fourier theorem for $\mathrm{SO}(3)$, we use the invariance of the integration measure dR: $\int_{\mathrm{SO}(3)} f(R'R)dR = \int_{\mathrm{SO}(3)} f(R)dR$.

With these facts understood, we can proceed to derive:

$$\begin{aligned}
\widehat{\psi \star f}^l &= \int_{\mathrm{SO}(3)} (\psi \star f)(R)\overline{D^l(R)}dR \\
&= \int_{\mathrm{SO}(3)}\int_{\mathrm{SO}(3)} \psi(R^{-1}R')f(R')dR'\overline{D^l(R)}dR \\
&= \int_{\mathrm{SO}(3)}\int_{\mathrm{SO}(3)} \psi(R^{-1})f(R')\overline{D^l(R'R)}dR'dR \\
&= \int_{\mathrm{SO}(3)} f(R')\overline{D^l(R')}dR' \int_{\mathrm{SO}(3)} \psi(R^{-1})\overline{D^l(R)}dR \\
&= \int_{\mathrm{SO}(3)} f(R')\overline{D^l(R')}dR' \int_{\mathrm{SO}(3)} \psi(R)\overline{D^l(R)}^{\dagger}dR \\
&= \hat{f}^l\ \hat{\psi}^{l\dagger}
\end{aligned} \tag{23}$$

So the $\mathrm{SO}(3)$-Fourier transform of the $\mathrm{SO}(3)$ convolution of ψ and f is equal to the matrix product of the $\mathrm{SO}(3)$-Fourier transforms $\hat{f}$ and $\hat{\psi}$.

For the sphere, we can derive an analogous transform that is sometimes called the spherical harmonics transform. The spherical harmonics $Y^l_m : S^2 \to \mathbb{C}$ are a complete orthogonal family of functions. The spherical harmonics are related to the Wigner D functions by the relation $D^l_{mn}(\alpha, \beta, \gamma) = Y^l_m(\alpha, \beta)e^{in\gamma}$, so that $Y^l_m(\alpha, \beta) = D^l_{m0}(\alpha, \beta, 0)$.

The S^2 convolution of f_1 and f_2 is equivalent to the $\mathrm{SO}(3)$ convolution of the associated right-invariant functions $\bar{f}_1, \bar{f}_2$ (see Appendix A):

$$\begin{aligned}
[f_1 \star f_2](R) &= \int_{S^2} f_1(R^{-1}x) f_2(x) dx \\
&= \int_{\mathrm{SO}(2)} \int_{S^2} f_1(R^{-1}x) f_2(x) dx dh \\
&= \int_{\mathrm{SO}(3)} \bar{f}_1(R^{-1}R') \bar{f}_2(R') dR' \\
&= [\bar{f}_1 \star \bar{f}_2](R)
\end{aligned} \tag{24}$$

The Fourier transform of a right invariant function on $\mathrm{SO}(3)$ equals

$$\begin{aligned}
[\mathcal{F}\bar{f}]^l_{mn} &= \int_0^{2\pi} \frac{d\alpha}{2\pi} \int_0^{\pi} \frac{d\beta \sin\beta}{2} \int_0^{2\pi} \frac{d\gamma}{2\pi} \bar{f}(\alpha,\beta,\gamma) \overline{D^l_{mn}(\alpha,\beta,\gamma)} \\
&= \int_0^{2\pi} \frac{d\alpha}{2\pi} \int_0^{\pi} \frac{d\beta \sin\beta}{2} f(\alpha,\beta) \int_0^{2\pi} \frac{d\gamma}{2\pi} \overline{D^l_{mn}(\alpha,\beta,\gamma)} \\
&= \delta_{n0} \int_0^{2\pi} \frac{d\alpha}{2\pi} \int_0^{\pi} \frac{d\beta \sin\beta}{2} f(\alpha,\beta) \overline{D^l_{m0}(\alpha,\beta,0)} \\
&= \delta_{n0} \int_{S^2} f(x) \overline{Y^l_m(x)} dx
\end{aligned} \tag{25}$$

So we can think of the S^2 Fourier transform of a function on S^2 as the $n = 0$ column of the $\mathrm{SO}(3)$ Fourier transform of the associated right-invariant function. This is a beautiful result that we have not been able to find a reference for, though it seems likely that it has been observed before.

Adversarial Examples that Fool both Computer Vision and Time-Limited Humans

Gamaleldin F. Elsayed[*]
Google Brain
gamaleldin.elsayed@gmail.com

Shreya Shankar
Stanford University

Brian Cheung
UC Berkeley

Nicolas Papernot
Pennsylvania State University

Alex Kurakin
Google Brain

Ian Goodfellow
Google Brain

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Abstract

Machine learning models are vulnerable to **adversarial examples**: small changes to images can cause computer vision models to make mistakes such as identifying a school bus as an ostrich. However, it is still an open question whether humans are prone to similar mistakes. Here, we address this question by leveraging recent techniques that transfer adversarial examples from computer vision models with known parameters and architecture to other models with unknown parameters and architecture, and by matching the initial processing of the human visual system. We find that adversarial examples that strongly transfer across computer vision models influence the classifications made by time-limited human observers.

1 Introduction

Machine learning models are easily fooled by adversarial examples: inputs optimized by an adversary to produce an incorrect model classification [3, 39]. In computer vision, an adversarial example is usually an image formed by making small perturbations to an example image. Many algorithms for constructing adversarial examples [13, 24, 27, 33, 39] rely on access to both the architecture and the parameters of the model to perform gradient-based optimization on the input. Without similar access to the brain, these methods do not seem applicable to constructing adversarial examples for humans.

One interesting phenomenon is that adversarial examples often transfer from one model to another, making it possible to attack models that an attacker has no access to [26, 39]. This naturally raises the question of whether humans are susceptible to these adversarial examples. Clearly, humans are prone to many cognitive biases and optical illusions [17], but these generally do not resemble small perturbations of natural images, nor are they currently generated by optimization of a ML loss function. Thus the current understanding is that this class of transferable adversarial examples has no effect on human visual perception, yet no thorough empirical investigation has yet been performed.

A rigorous investigation of the above question creates an opportunity both for machine learning to gain knowledge from neuroscience, and for neuroscience to gain knowledge from machine learning. Neuroscience has often provided existence proofs for machine learning—before we had working object recognition algorithms, we hypothesized it should be possible to build them because the human brain can recognize objects. See Hassabis et al. [15] for a review of the influence of neuroscience on artificial intelligence. If we knew conclusively that the human brain could resist a certain class of adversarial examples, this would provide an existence proof for a similar mechanism in machine learning security. If we knew conclusively that the brain can be fooled by adversarial examples, then machine learning security research should perhaps shift its focus from designing models that

[*]Work done as a member of the Google AI Residency program (g.co/airesidency).

are robust to adversarial examples [5, 13, 19, 27, 32, 39, 40, 42] to designing systems that are secure despite including non-robust machine learning components. Likewise, if adversarial examples developed for computer vision affect the brain, this phenomenon discovered in the context of machine learning could lead to a better understanding of brain function.

In this work, we construct adversarial examples that transfer from computer vision models to the human visual system. In order to successfully construct these examples and observe their effect, we leverage three key ideas from machine learning, neuroscience, and psychophysics. First, we use the recent **black box** adversarial example construction techniques that create adversarial examples for a target model without access to the model's architecture or parameters. Second, we adapt machine learning models to mimic the initial visual processing of humans, making it more likely that adversarial examples will transfer from the model to a human observer. Third, we evaluate classification decisions of human observers in a time-limited setting, so that even subtle effects on human perception are detectable. By making image presentation sufficiently brief, humans are unable to achieve perfect accuracy even on clean images, and small changes in performance lead to more measurable changes in accuracy. Additionally, a brief image presentation limits the time in which the brain can utilize recurrent and top-down processing pathways [34], and is believed to make the processing in the brain more closely resemble that in a feedforward artificial neural network.

We find that adversarial examples that transfer across computer vision models *do* successfully influence the perception of human observers, thus uncovering a new class of illusions that are shared between computer vision models and the human brain.

2 Background and Related Work

2.1 Adversarial Examples

Goodfellow et al. [12] define adversarial examples as "inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake." In the context of visual object recognition, adversarial examples are images usually formed by applying a small perturbation to a naturally occurring image in a way that breaks the predictions made by a machine learning classifier. See Figure 1a for a canonical example where adding a small perturbation to an image of a panda causes it to be misclassified as a gibbon. This perturbation is small enough to be imperceptible (i.e., it cannot be saved in a standard png file that uses 8 bits because the perturbation is smaller than $1/255$ of the pixel dynamic range). This perturbation relies on carefully chosen structure based on the parameters of the neural network—but when magnified to be perceptible, human observers cannot recognize any meaningful structure. Note that adversarial examples also exist in other domains like malware detection [14], but we focus here on image classification tasks.

Two aspects of the definition of adversarial examples are particularly important for this work:

1. Adversarial examples are designed to cause a *mistake*. They are not (as is commonly misunderstood) defined to differ from human judgment. If adversarial examples were defined by deviation from human output, it would by definition be impossible to make adversarial examples for humans. On some tasks, like predicting whether input numbers are prime, there is a clear objectively correct answer, and we would like the model to get the correct answer, not the answer provided by humans (time-limited humans are probably not very good at guessing whether numbers are prime). It is challenging to define what constitutes a mistake for visual object recognition. After adding a perturbation to an image it likely no longer corresponds to a photograph of a real physical scene. Furthermore, it is philosophically difficult to define the real object class for an image that is not a picture of a real object. In this work, we assume that an adversarial image is misclassified if the output label differs from the human-provided label of the clean image that was used as the starting point for the adversarial image. We make small adversarial perturbations and we assume that these small perturbations are insufficient to change the true class.

2. Adversarial examples are not (as is commonly misunderstood) defined to be imperceptible. If this were the case, it would be impossible by definition to make adversarial examples for humans, because changing the human's classification would constitute a change in what the human perceives (e.g., see Figure 1b,c).

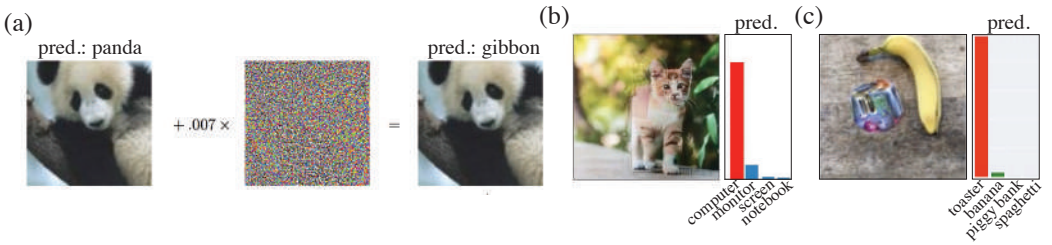


Figure 1: **Adversarial examples optimized on more models / viewpoints sometimes appear more meaningful to humans.** This observation is a clue that machine-to-human transfer may be possible. (a) A canonical example of an adversarial image reproduced from [13]. This adversarial attack has moderate but limited ability to fool the model after geometric transformations or to fool models other than the model used to generate the image. (b) An adversarial attack causing a cat image to be labeled as a computer while being robust to geometric transformations, adopted from [1]. Unlike the attack in a, the image contains features that seem semantically computer-like to humans. (c) An adversarial patch that causes images to be labeled as a toaster, optimized to cause misclassification from multiple viewpoints, reproduced from [4]. Similar to b, the patch contains features that appear toaster-like to a human.

2.1.1 Clues that Transfer to Humans is Possible

Some observations give clues that transfer to humans may be possible. Adversarial examples are known to transfer across machine learning models, which suggest that these adversarial perturbations may carry information about target adversarial classes. Adversarial examples that fool one model often fool another model with a different architecture [39], another model that was trained on a different training set [39], or even trained with a different algorithm [30] (e.g., adversarial examples designed to fool a convolution neural network may also fool a decision tree). The transfer effect makes it possible to perform black box attacks, where adversarial examples fool models that an attacker does not have access to [31, 39]. Kurakin et al. [24] found that adversarial examples transfer from the digital to the physical world, despite many transformations such as lighting and camera effects that modify their appearance when they are photographed in the physical world. Liu et al. [26] showed that the transferability of an adversarial example can be greatly improved by optimizing it to fool *many* machine learning models rather than one model: an adversarial example that fools five models used in the optimization process is more likely to fool an arbitrary sixth model.

Moreover, recent studies on stronger adversarial examples that transfer across multiple settings have sometimes produced adversarial examples that appear more meaningful to human observers. For instance, a cat adversarially perturbed to resemble a computer [2] while transfering across geometric transformations develops features that appear computer-like (Figure 1b), and the 'adversarial toaster' from Brown et al. [4] possesses features that seem toaster-like (Figure 1c). This development of human-meaningful features is consistent with the adversarial example carrying true feature information and thus coming closer to fooling humans, if we acounted for the notable differences between humans visual processing and computer vision models (see section 2.2.2)

2.2 Biological and Artificial Vision

2.2.1 Similarities

Recent research has found similarities in representation and behavior between deep convolutional neural networks (CNNs) and the primate visual system [6]. This further motivates the possibility that adversarial examples may transfer from computer vision models to humans. Activity in deeper CNN layers has been observed to be predictive of activity recorded in the visual pathway of primates [6, 43]. Reisenhuber and Poggio [36] developed a model of object recognition in cortex that closely resembles many aspects of modern CNNs. Kummerer et al. [21, 22] showed that CNNs are predictive of human gaze fixation. Style transfer [10] demonstrated that intermediate layers of a CNN capture notions of artistic style which are meaningful to humans. Freeman et al. [9] used representations in a CNN-like model to develop psychophysical metamers, which are indistinguishable to humans when viewed

briefly and with carefully controlled fixation. Psychophysics experiments have compared the pattern of errors made by humans, to that made by neural network classifiers [11, 35].

2.2.2 Notable Differences

Differences between machine and human vision occur early in the visual system. Images are typically presented to CNNs as a static rectangular pixel grid with constant spatial resolution. The primate eye on the other hand has an eccentricity dependent spatial resolution. Resolution is high in the fovea, or central $\sim 5^\circ$ of the visual field, but falls off linearly with increasing eccentricity [41]. A perturbation which requires high acuity in the periphery of an image, as might occur as part of an adversarial example, would be undetectable by the eye, and thus would have no impact on human perception. Further differences include the sensitivity of the eye to temporal as well as spatial features, as well as non-uniform color sensitivity [25]. Modeling the early visual system continues to be an area of active study [28, 29]. As we describe in section 3.1.2, we mitigate some of these differences by using a biologically-inspired image input layer.

Beyond early visual processing, there are more major computational differences between CNNs and the human brain. All the CNNs we consider are fully feedforward architectures, while the visual cortex has many times more feedback than feedforward connections, as well as extensive recurrent dynamics [29]. Possibly due to these differences in architecture, humans have been found experimentally to make classification mistakes that are qualitatively different than those made by deep networks [8]. Additionally, the brain does not treat a scene as a single static image, but actively explores it with saccades [18]. As is common in psychophysics experiments [20], we mitigate these differences in processing by limiting both the way in which the image is presented, and the time which the subject has to process it, as described in section 3.2.

3 Methods

Section 3.1 details our machine learning vision pipeline. Section 3.2 describes our psychophysics experiment to evaluate the impact of adversarial images on human subjects.

3.1 The Machine Learning Vision Pipeline

3.1.1 Dataset

In our experiment, we used images from ImageNet [7]. ImageNet contains 1,000 highly specific classes that typical people may not be able to identify, such as "Chesapeake Bay retriever". Thus, we combined some of these fine classes to form six coarse classes we were confident would be familiar to our experiment subjects ($\{$dog, cat, broccoli, cabbage, spider, snake$\}$). We then grouped these six classes into the following groups: (i) **Pets** group (dog and cat images); (ii) **Hazard** group (spider and snake images); (iii) **Vegetables** group (broccoli and cabbage images).

3.1.2 Ensemble of Models

We constructed an ensemble of k CNN models trained on ImageNet ($k = 10$). Each model is an instance of one of these architectures: Inception V3, Inception V4, Inception ResNet V2, ResNet V2 50, ResNet V2 101, and ResNet V2 152 [16, 37, 38]. To better match the initial processing of human visual system, we prepend each model with a retinal layer, which pre-processes the input to incorporate some of the transformations performed by the human eye. In that layer, we perform an eccentricity dependent blurring of the image to approximate the input which is received by the visual cortex of human subjects through their retinal lattice. The details of this retinal layer are described in Appendix B. We use eccentricity-dependent spatial resolution measurements (based on the macaque visual system) [41], along with the known geometry of the viewer and the screen, to determine the degree of spatial blurring at each image location. This limits the CNN to information which is also available to the human visual system. The layer is fully differentiable, allowing gradients to backpropagate through the network when running adversarial attacks. Further details of the models and their classification performance are provided in Appendix E.

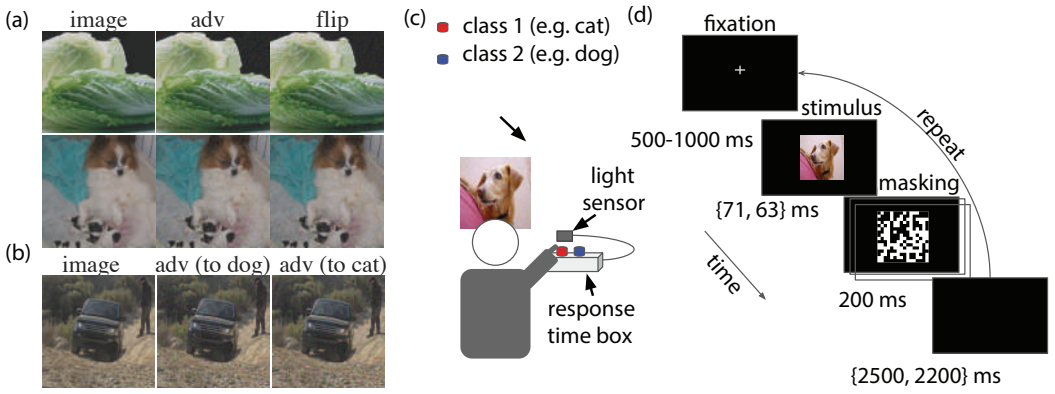


Figure 2: **Experiment setup and task.** (a) examples images from the conditions (`image`, `adv`, and `flip`). Top: `adv` targeting broccoli class. bottom: `adv` targeting cat class. See definition of conditions at Section 3.2.2. (b) example images from the `false` experiment condition. (c) Experiment setup and recording apparatus. (d) Task structure and timings. The subject is asked to repeatedly identify which of two classes (e.g. dog vs. cat) a briefly presented image belongs to. The image is either adversarial, or belongs to one of several control conditions. See Section 3.2 for details.

3.1.3 Generating Adversarial Images

For a given image group, we wish to generate targeted adversarial examples that strongly transfer across models. This means that for a class pair (A, B) (e.g., A: cats and B: dogs), we generate adversarial perturbations such that models will classify perturbed images from A as B; similarly, we perturbed images from B to be classified as A. A different perturbation is constructed for each image; however, the ℓ_∞ norm of all perturbations are constrained to be equal to a fixed ϵ.

Formally: given a classifier k, which assigns probability $P(y \mid X)$ to each coarse class y given an input image X, a specified target class y_{target} and a maximum perturbation ϵ, we want to find the image X_{adv} that minimizes $-\log(P(y_{\text{target}} \mid X_{adv}))$ with the constraint that $||X_{adv} - X||_\infty = \epsilon$. See Appendix C for details on computing the coarse class probabilities $P(y \mid X)$. With the classifier's parameters, we can perform iterated gradient descent on X in order to generate our X_{adv} (see Appendix D). This iterative approach is commonly employed to generate adversarial images [24].

3.2 Human Psychophysics Experiment

38 subjects with normal or corrected vision participated in the experiment. Subjects gave informed consent to participate, and were awarded a reasonable compensation for their time and effort[2].

3.2.1 Experimental Setup

Subjects sat on a fixed chair 61cm away from a high refresh-rate computer screen (ViewSonic XG2530) in a room with dimmed light (Figure 2c). Subjects were asked to classify images that appeared on the screen to one of two classes (two alternative forced choice) by pressing buttons on a response time box (LOBES v5/6:USTC) using two fingers on their right hand. The assignment of classes to buttons was randomized for each experiment session. Each trial started with a fixation cross displayed in the middle of the screen for $500 - 1000$ ms, instructing subjects to direct their gaze to the fixation cross (Figure 2d). After the fixation period, an image of size 15.24 cm $\times$ 15.24 cm (14.2° visual angle) was presented briefly at the center of the screen for a period of 63 ms (71 ms for some sessions). The image was followed by a sequence of ten high contrast binary random masks, each displayed for 20 ms (see example in Figure 2d). Subjects were asked to classify the object in the image (e.g., cat vs. dog) by pressing one of two buttons starting at the image presentation time and lasting until 2200 ms (or 2500 ms for some sessions) after the mask was turned off. The waiting period to start the next trial was of the same duration whether subjects responded quickly or slowly.

[2]The study was granted an Institutional Review Board (IRB) exemption by an external, independent, ethics board (Quorum review ID 33016).

Realized exposure durations were ± 4ms from the times reported above, as measured by a photodiode and oscilloscope in a separate test experiment. Each subject's response time was recorded by the response time box relative to the image presentation time (monitored by a photodiode). In the case where a subject pressed more than one button in a trial, only the class corresponding to their first choice was considered. Each subject completed between 140 and 950 trials.

3.2.2 Experiment Conditions

Each experimental session included only one of the image groups (Pets, Vegetables or Hazard). For each group, images were presented in one of four conditions as follows:

- `image`: images from the ImageNet training set (rescaled to the $[40, 255-40]$ range to avoid clipping when adversarial perturbations are added; see Figure 2a left)).
- `adv`: we added adversarial perturbation δ_{adv} to `image`, crafted to cause machine learning models to misclassify `adv` as the opposite class in the group (e.g., if `image` was originally a cat, we perturbed the image to be classified as a dog). We used a perturbation size large enough to be noticeable by humans on the computer screen but small with respect to the image intensity scale ($\epsilon = 32$; see Figure 2a middle). In other words, we chose ϵ to be large (to improve the chances of adversarial examples transfer to time-limited human) but kept it small enough that the perturbations are class-preserving (as judged by a no-limit human).
- `flip`: similar to `adv`, but the adversarial perturbation (δ_{adv}) is flipped vertically before being added to `image`. This is a control condition, chosen to have nearly identical perturbation statistics to the `adv` condition (see Figure 2a right). We include this condition because if adversarial perturbations *reduce the accuracy* of human observers, this could just be because the perturbations degrade the image quality.
- `false`: in this condition, subjects are forced to make a mistake. To show that adversarial perturbations *actually control the chosen class*, we include this condition where neither of the two options available to the subject is correct, so their accuracy is always zero. We test whether adversarial perturbations can influence which of the two wrong choices they make. We show a random image from an ImageNet class other than the two classes in the group, and adversarially perturb it toward one of the two classes in the group. The subject must then choose from these two classes. For example, we might show an airplane adversarially perturbed toward the dog class, while a subject is in a session classifying images as cats or dogs. We used a slightly larger perturbation in this condition ($\epsilon = 40$; see Figure 2b).

The conditions (`image`, `adv`, `flip`) are ensured to have balanced number of trials within a session by either uniformly sampling the condition type in some of the sessions or randomly shuffling a sequence with identical trial counts for each condition in other sessions. The number of trials for each class in the group was also constrained to be equal. Similarly for the `false` condition the number of trials adversarially perturbed towards class 1 and class 2 were balanced for each session. To reduce subjects using strategies based on overall color or brightness distinctions between classes, we pre-filtered the dataset to remove images that showed an obvious effect of this nature. Notably, in the pets group we excluded images that included large green lawns or fields, since in almost all cases these were photographs of dogs. See Appendix F for images used in the experiment for each coarse class. For example images for each condition, see Figures Supp.2 through Supp.5.

4 Results

4.1 Adversarial Examples Transfer to Computer Vision Models

We first assess the transfer of our constructed images to two test models that were not included in the ensemble used to generate adversarial examples. These test models are an adversarially trained Inception V3 model [23] and a ResNet V2 50 model. Both models perform well ($> 75\%$ accuracy) on clean images. Attacks in the `adv` and `false` conditions succeeded against the test models between 57% and 89% of the time, depending on image class and experimental condition. The `flip` condition changed the test model predictions on fewer than 1.5% of images in all conditions, validating its use as a control. See Tables Supp.3 - Supp.6 for accuracy and attack success measurements on both train and test models for all experimental conditions.

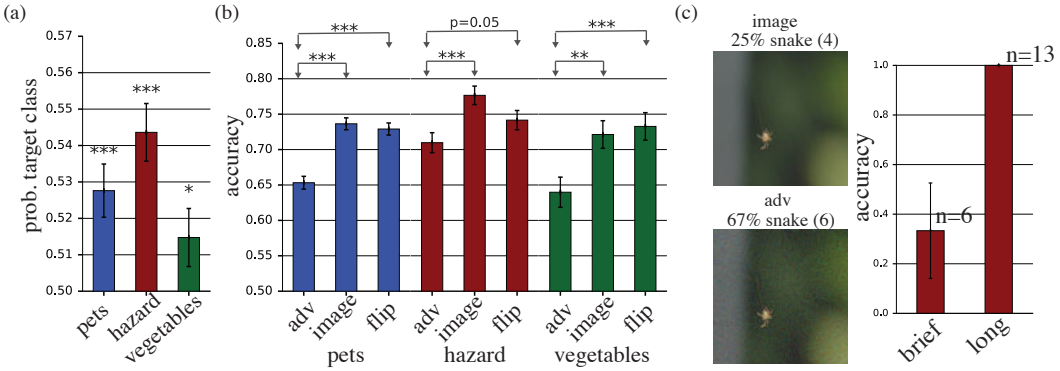


Figure 3: **Adversarial images transfer to humans.** (a) By adding adversarial perturbations to an image, we are able to bias which of two incorrect choices subjects make. Plot shows probability of choosing the adversarially targeted class when the true image class is not one of the choices that subjects can report (`false` condition), estimated by averaging the responses of all subjects (two-tailed t-test relative to chance level 0.5). (b) Adversarial images cause more mistakes than either clean images or images with the adversarial perturbation flipped vertically before being applied. Plot shows probability of choosing the true image class, when this class is one of the choices that subjects can report, averaged across all subjects. Accuracy is significantly less than 1 even for clean images due to the brief image presentation time. (error bars $\pm$ SE; *: $p < 0.05$; **: $p < 0.01$; ***: $p < 0.001$) (c) A spider image that time-limited humans frequently perceived as a snake (top parentheses: number of subjects tested on this image). right: accuracy on this adversarial image when presented briefly compared to when presented for long time (long presentation is based on a post-experiment email survey of 13 participants).

4.2 Adversarial Examples Transfer to Humans

We now show that adversarial examples transfer to time-limited humans. One could imagine that adversarial examples merely degrade image quality or discard information, thus increasing error rate. To rule out this possibility, we begin by showing that for a fixed error rate (in a setting where the human is forced to make a mistake), adversarial perturbations influence the human choice among two incorrect classes. Then, we demonstrate that adversarial examples increase the error rate.

4.2.1 Influencing the Choice between two Incorrect Classes

As described in Section 3.2.2, we used the `false` condition to test whether adversarial perturbations can influence which of two incorrect classes a subject chooses (see example images in Figure Supp.2).

We measured our effectiveness at changing the perception of subjects using the rate at which subjects reported the adversarially targeted class. If the adversarial perturbation were completely ineffective we would expect the choice of targeted class to be uncorrelated with the subject's reported class. The average rate at which the subject chooses the target class metric would be 0.5 as each `false` image can be perturbed to class 1 or class 2 in the group with equal probability. Figure 3a shows the probability of choosing the target class averaged across all subjects for all the three experiment groups. In all cases, the probability was significantly above the chance level of 0.5. This demonstrates that the adversarial perturbations generated using CNNs biased human perception towards the targeted class. This effect was stronger for the the hazard, then pets, then vegetables group. This difference in probability among the class groups was significant ($p < 0.05$; Pearson Chi-2 GLM test).

We also observed a significant difference in the mean response time between the class groups ($p < 0.001$; one-way ANOVA test; see Figure 4a). Interestingly, the response time pattern across image groups (Figure 4a)) was inversely correlated to the perceptual bias pattern (Figure 3a)) (Pearson correlation $= -1$, $p < 0.01$; two-tailed Pearson correlation test). In other words, subjects made quicker decisions for the hazard group, then pets group, and then vegetables group. This is consistent with subjects being more confident in their decision when the adversarial perturbation was more

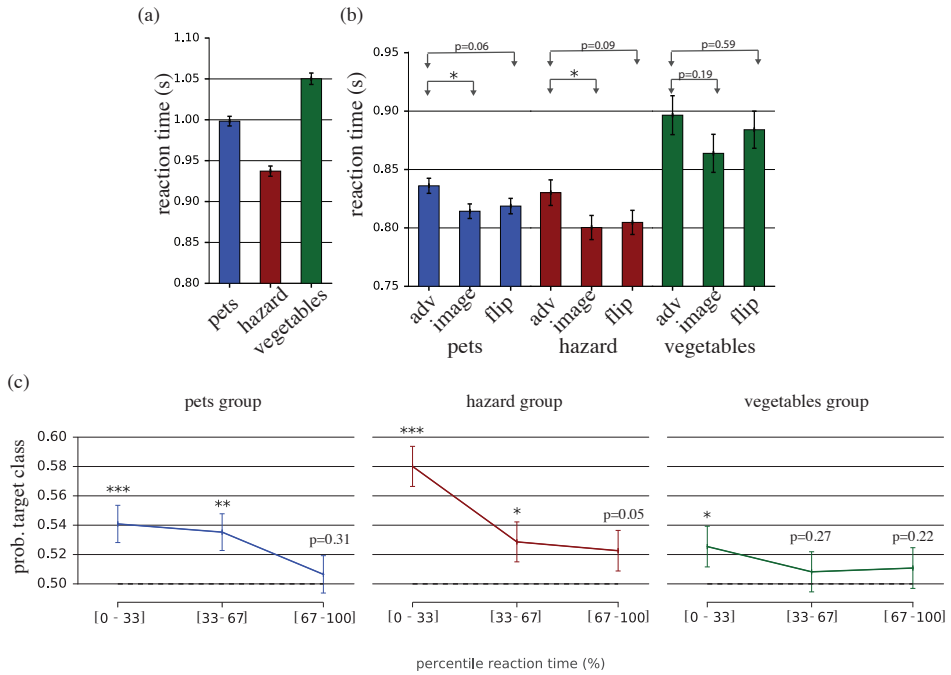


Figure 4: **Adversarial images effect human response time.** (a) Average response time to `false` images. (b) Average response time for `adv`, `image`, and `flip` conditions (error bars $\pm$ SE; * reflects $p < 0.05$; two sample two-tailed t-test). In all three stimulus groups, there was a trend towards slower response times in the `adv` condition than in either control group. (c) Probability of choosing the adversarially targeted class in the `false` condition, estimated by averaging the responses of all subjects (two-tailed t-test relative to chance level 0.5; error bars $\pm$ SE; *: $p < 0.05$; **: $p < 0.01$; ***: $p < 0.001$). The probability of choosing the targeted label is computed by binning trials within percentile reaction time ranges (0-33 percentile, 33-67 percentile, and 67-100 percentile). The bias relative to chance level of 0.5 is significant when people reported their decision quickly (when they may have been more confident), but not significant when they reported their decision more slowly. As discussed in Section 4.2.2, differing effect directions in (b) and (c) may be explained by adversarial perturbations decreasing decision confidence in the `adv` condition, and increasing decision confidence in the `false` condition.

successful in biasing subjects perception. This inverse correlation between attack success and response time was observed within group, as well as between groups (Figure 4).

4.2.2 Adversarial Examples Increase Human Error Rate

We demonstrated that we are able to bias human perception to a target class when the true class of the image is not one of the options that subjects can choose. Now we show that adversarial perturbations can be used to cause the subject to choose an incorrect class even though the correct class is an available response. As described in Section 3.2.2, we presented `image`, `flip`, and `adv`.

Most subjects had lower accuracy in `adv` than `image` (Table Supp.1). This is also reflected on the average significantly lower accuracy across all subjects for the `adv` than `image` (Figure 3b).

The above result may simply imply that the signal to noise ratio in the adversarial images is lower than that of clean images. While this issue is partially addressed with the `false` experiment results in Section 4.2.1, we additionally tested accuracy on `flip` images. This control case uses perturbations with identical statistics to `adv` up to a flip of the vertical axis. However, this control breaks the pixel-to-pixel correpsondence between the adversarial perturbation and the image. The majority of subjects had lower accuracy in the `adv` condition than in the `flip` condition (Table Supp.1). When averaging across all trials, this effect was very significant for the pets and vegetables group

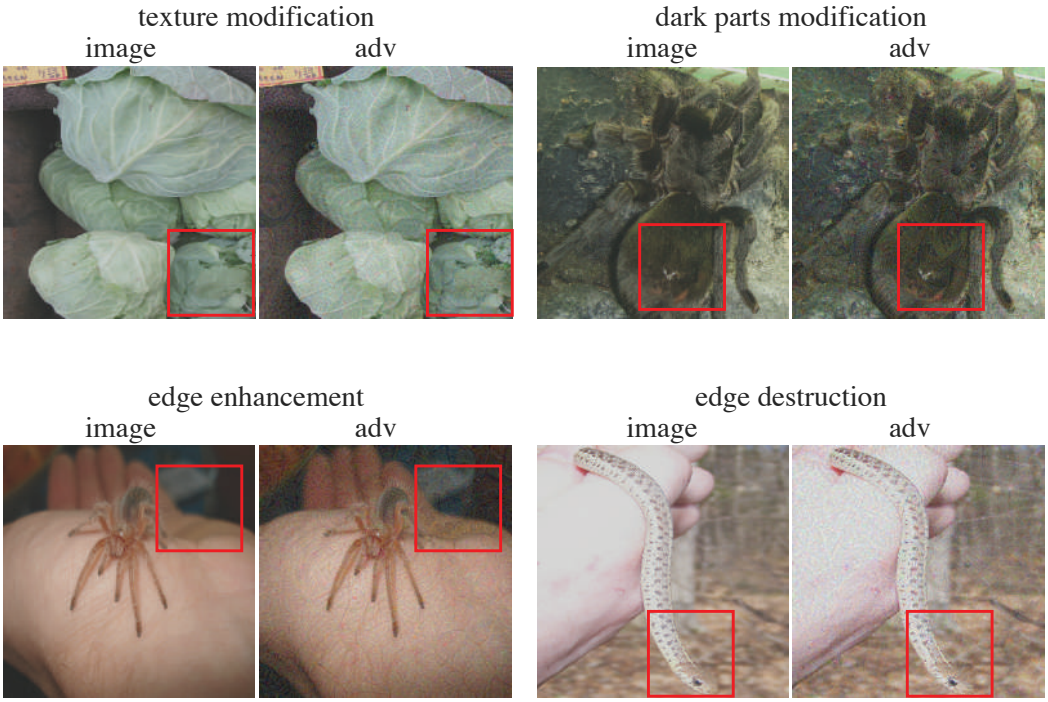


Figure 5: **Examples of the types of manipulations performed by the adversarial attack.** See Figures Supp.3 through Supp.5 for additional examples of adversarial images. Also see Figure Supp.2 for adversarial examples from the `false` condition.

($p < 0.001$), and less significant for the hazard group ($p = 0.05$) (Figure 3b). These results suggest that the direction of the adversarial image perturbation, in combination with a specific image, is perceptually relevant to features that the human visual system uses to classify objects. These findings thus give evidence that strong black box adversarial attacks can transfer from CNNs to humans, and show remarkable similarities between failure cases of CNNs and human vision.

In all cases, the average response time was longer for the `adv` condition relative to the other conditions (Figure 4b), though this result was only statistically significant for two comparisons. If this trend remains predictive, it would seem to contradict the case when we presented `false` images (Figure 4a). One interpretation is that in the `false` case, the transfer of adversarial features to humans was accompanied by more confidence, whereas here the transfer was accompanied by less confidence, possibly due to competing adversarial and true class features in the `adv` condition.

5 Discussion

Our results invite several questions that we discuss briefly.

5.1 Have we actually fooled human observers or did we change the true class?

One might naturally wonder whether we have fooled the human observer or whether we have replaced the input image with an image that actually belongs to a different class. In our work, the perturbations we made were small enough that they generally do not change the output class for a human who has no time limit (the reader may verify this by observing Figures 2a,b, 3c, and Supp.2 through Supp.5).

. We can thus be confident that we did not change the true class of the image, and that we really did fool the time-limited human. Future work aimed at fooling humans with no time-limit will need to tackle the difficult problem of obtaining a better ground truth signal than visual labeling by humans.

5.2 How do the adversarial examples work?

We did not design controlled experiments to prove that the adversarial examples work in any specific way, but we informally observed a few apparent patterns illustrated in Figure 5: disrupting object edges, especially by mid-frequency modulations perpendicular to the edge; enhancing edges both by increasing contrast and creating texture boundaries; modifying texture; and taking advantage of dark regions in the image, where the perceptual magnitude of small ϵ perturbations can be larger.

5.3 What are the implications for machine learning security and society?

The fact that our transfer-based adversarial examples fool time-limited humans but not no-limit humans suggests that the lateral and top-down connections used by the no-limit human are relevant to human robustness to adversarial examples. This suggests that machine learning security research should explore the significance of these top-down or lateral connections further. One possible explanation for our observation is that no-limit humans are fundamentally more robust to adversarial example and achieve this robustness via top-down or lateral connections. If this is the case, it could point the way to the development of more robust machine learning models. Another possible explanation is that no-limit humans remain highly vulnerable to adversarial examples but adversarial examples do not transfer from feed-forward networks to no-limit humans because of these architectural differences.

Our results suggest that there is a risk that imagery could be manipulated to cause human observers to have unusual reactions; for example, perhaps a photo of a politician could be manipulated in a way that causes it to be perceived as unusually untrustworthy or unusually trustworthy in order to affect the outcome of an election.

5.4 Future Directions

In this study, we designed a procedure that according to our hypothesis would transfer adversarial examples to humans. An interesting set of questions relates to how sensitive that transfer is to different elements of our experimental design. For example: How does transfer depend on ϵ? Was model ensembling crucial to transfer? Can the retinal preprocessing layer be removed? We suspect that retinal preprocessing and ensembling are both important for transfer to humans, but that ϵ could be made smaller. See Figure Supp.1 for a preliminary exploration of these questions.

6 Conclusion

In this work, we showed that adversarial examples based on perceptible but class-preserving perturbations that fool multiple machine learning models also fool time-limited humans. Our findings demonstrate striking similarities between convolutional neural networks and the human visual system. We expect this observation to lead to advances in both neuroscience and machine learning research.

Acknowledgements

We are grateful to Ari Morcos, Bruno Olshausen, David Sussillo, Hanlin Tang, John Cunningham, Santani Teng, and Daniel Yamins for useful discussions. We also thank Dan Abolafia Simon Kornblith, Katherine Lee, Niru Maheswaranathan, Catherine Olsson, David Sussillo, and Santani Teng, for helpful feedback on the manuscript. We thank Google Brain residents for useful feedback on the work. We also thank Deanna Chen, Leslie Philips, Sally Jesmonth, Phing Turner, Melissa Strader, Lily Peng, and Ricardo Prada for assistance with IRB and experiment setup.

References

[1] Anish Athalye. Robust adversarial examples, 2017.

[2] Anish Athalye and Ilya Sutskever. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In

Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III, pages 387–402, 2013.

[4] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[5] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. *International Conference on Learning Representations*, 2018. accepted as poster.

[6] Charles F Cadieu, Ha Hong, Daniel LK Yamins, Nicolas Pinto, Diego Ardila, Ethan A Solomon, Najib J Majaj, and James J DiCarlo. Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS computational biology*, 10(12):e1003963, 2014.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[8] Miguel P Eckstein, Kathryn Koehler, Lauren E Welbourne, and Emre Akbas. Humans, but not deep neural networks, often miss giant targets in scenes. *Current Biology*, 27(18):2827–2832, 2017.

[9] Jeremy Freeman and Eero P Simoncelli. Metamers of the ventral stream. *Nature neuroscience*, 14(9):1195, 2011.

[10] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[11] Robert Geirhos, David HJ Janssen, Heiko H Schütt, Jonas Rauber, Matthias Bethge, and Felix A Wichmann. Comparing deep neural networks against humans: object recognition when the signal gets weaker. *arXiv preprint arXiv:1706.06969*, 2017.

[12] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Yan Duan, Pieter Abbeel, and Jack Clark. Attacking machine learning with adversarial examples, 2017.

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[14] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick D. McDaniel. Adversarial examples for malware detection. In *ESORICS 2017*, pages 62–79, 2017.

[15] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks. *ArXiv e-prints*, March 2016.

[17] James M Hillis, Marc O Ernst, Martin S Banks, and Michael S Landy. Combining sensory information: mandatory fusion within, but not between, senses. *Science*, 298(5598):1627–1630, 2002.

[18] Michael Ibbotson and Bart Krekelberg. Visual perception and saccadic eye movements. *Current opinion in neurobiology*, 21(4):553–558, 2011.

[19] J Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.

[20] GYULA KovAcs, Rufin Vogels, and Guy A Orban. Cortical correlate of pattern backward masking. *Proceedings of the National Academy of Sciences*, 92(12):5587–5591, 1995.

[21] Matthias Kümmerer, Lucas Theis, and Matthias Bethge. Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet. *arXiv preprint arXiv:1411.1045*, 2014.

[22] Matthias Kümmerer, Tom Wallis, and Matthias Bethge. Deepgaze ii: Predicting fixations from deep features over time and tasks. *Journal of Vision*, 17(10):1147–1147, 2017.

[23] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial Machine Learning at Scale. *ArXiv e-prints*, November 2016.

[24] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR'2017 Workshop*, 2016.

[25] Michael F Land and Dan-Eric Nilsson. *Animal eyes*. Oxford University Press, 2012.

[26] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[27] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[28] Lane McIntosh, Niru Maheswaranathan, Aran Nayebi, Surya Ganguli, and Stephen Baccus. Deep learning models of the retinal response to natural scenes. In *Advances in neural information processing systems*, pages 1369–1377, 2016.

[29] Bruno A Olshausen. 20 years of learning about vision: Questions answered, questions unanswered, and questions not yet asked. In *20 Years of Computational Neuroscience*, pages 243–270. Springer, 2013.

[30] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[31] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[32] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.

[33] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015.

[34] Mary C Potter, Brad Wyble, Carl Erick Hagmann, and Emily S McCourt. Detecting meaning in rsvp at 13 ms per picture. *Attention, Perception, & Psychophysics*, 76(2):270–279, 2014.

[35] Rishi Rajalingham, Elias B. Issa, Pouya Bashivan, Kohitij Kar, Kailyn Schmidt, and James J DiCarlo. Large-scale, high-resolution comparison of the core visual object recognition behavior of humans, monkeys, and state-of-the-art deep artificial neural networks. *bioRxiv*, 2018.

[36] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019, 1999.

[37] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *ArXiv e-prints*, February 2016.

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. *ArXiv e-prints*, December 2015.

[39] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[40] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. Ensemble Adversarial Training: Attacks and Defenses. *ArXiv e-prints*, May 2017.

[41] D. C. Van Essen and C. H Anderson. Information processing strategies and pathways in the primate visual system. In Zornetzer S. F., Davis J. L., Lau C., and McKenna T., editors, *An introduction to neural and electronic networks*, page 45–76, San Diego, CA, 1995. Academic Press.

[42] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[43] Daniel L. K. Yamins and James J. DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19:356–365, 2016.

Supplemental material

A Supplementary Figures and Tables

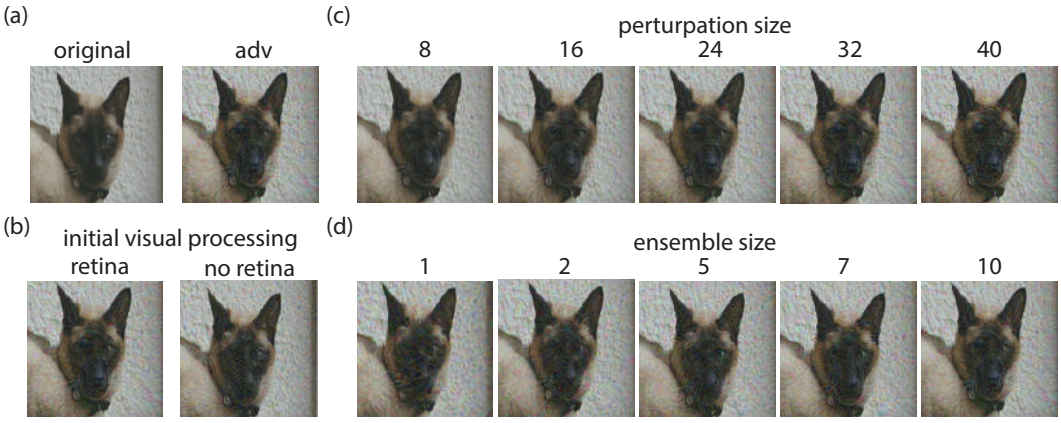


Figure Supp.1: **Intuition on factors contributing to the transfer to humans.** To give some intuition on the factors contributing to transfer, we examine a cat image from ImageNet (*(a)* left) that is already perceptually close to the target adversarial dog class, making the impact of subtle adversarial effects more obvious even on long observation (*(a)* right). Note that this image was not used in the experiment, and that typical images in the experiment *did not* fool unconstrained humans. *(b)* shows adversarial images with different perturbation sizes ranging from $\epsilon = 8$ to $\epsilon = 40$. Even smaller perturbation $\epsilon = 8$ make the adversarial image perceptually more similar to a dog image, which suggests that transfer to humans may be robust to small ϵ. *(c)* Investigation of the importance of matching initial visual processing. The adversarial image on the left is similar to a dog, while removing the retina layer leads to an image which is less similar to a dog. This suggests that matching initial processing is an important factor in transferring adversarial examples to humans. *(d)* Investigation of the importance of the number of models in the ensemble. We generated adversarial images with $\epsilon = 32$ using an ensemble of size 1– 10 models. One can see that adversarial perturbations become markedly less similar to a dog class as the number of models in the ensemble is reduced. This supports the importance of ensembling to the transfer of adversarial examples to humans.

Table Supp.1: **Adversarial examples transfer to humans.** Number of subjects that reported the correct class of images in the `adv` condition with lower mean accuracy compared to their mean accuracy in the `image` and `flip` conditions.

Group	`adv` <`image`	`adv` < `flip`	**total**
pets	29	22	35
hazard	19	16	24
vegetables	21	23	32

Table Supp.2: **Accuracy of models on ImageNet validation set.** * models trained on ImageNet with retina layer pre-pended and with train data augmented with rescaled images in the range of $[40, 255 - 40]$; ** model trained with adversarial examples augmented data. First ten models are models used in the adversarial training ensemble. Last two models are models used to test the transferability of adversarial examples.

Model	**Top-1 accuracy**
Resnet V2 101	0.77
Resnet V2 101*	0.7205
Inception V4	0.802
Inception V4*	0.7518
Inception Resnet V2	0.804
Inception Resnet V2*	0.7662
Inception V3	0.78
Inception V3*	0.7448
Resnet V2 152	0.778
Resnet V2 50*	0.708
Resnet V2 50 (test)	0.756
Inception V3** (test)	0.776

Table Supp.3: **Accuracy of ensemble used to generate adversarial examples on images at different conditions.** * models trained on ImageNet with retina layer appended and with train data augmented with rescaled images in the range of $[40, 255 - 40]$; Numbers triplet reflects accuracy on images from pets, hazard, and vegetables groups, respectively.

Train Model	`adv` (%)	`flip` (%)
Resnet V2 101	0.0, 0.0, 0.0	95, 92, 91
Resnet V2 101*	0.0, 0.0, 0.0	87, 87, 77
Inception V4	0.0, 0.0, 0.0	96, 95, 86
Inception V4*	0.0, 0.0, 0.0	87, 87, 73
Inception Resnet V2	0.0, 0.0, 0.0	97, 95, 95
Inception Resnet V2*	0.0, 0.0, 0.0	87, 83, 73
Inception V3	0.0, 0.0, 0.0	97, 94, 89
Inception V3*	0.0, 0.0, 0.0	83, 86, 74
Resnet V2 152	0.0, 0.0, 0.0	96, 95, 91
Resnet V2 50*	0.0, 0.0, 0.0	82, 85, 81

Table Supp.4: **Accuracy of test models on images at different conditions.** ** model trained on both clean and adversarial images. Numbers triplet is accuracy on pets, hazard, and vegetables groups, respectively.

Model	`adv` (%)	`flip` (%)
Resnet V2 50	8.7, 9.4, 13	93, 91, 85
Inception V3**	6.0, 6.9, 17	95, 92, 94

Table Supp.5: **Attack success on model ensemble.** Same convention as Table Supp.3

Model	adv (%)	flip (%)
Resnet V2 101	100, 100, 100	2, 0, 0
Resnet V2 101*	100, 100, 100	3, 0, 0
Inception V4	100, 100, 100	1, 0, 1
Inception V4*	100, 100, 100	4, 1, 0
Inception Resnet V2	100, 100, 100	1, 0, 1
Inception Resnet V2*	100, 100, 100	5, 2, 0
Inception V3	100, 100, 100	1, 0, 0
Inception V3*	100, 100, 100	5, 1, 1
Resnet V2 152	100, 100, 100	1, 0, 0
Resnet V2 50*	100, 100, 100	3, 1, 0

Table Supp.6: **Attack success on test models.** ** model trained on both clean and adversarial images. Numbers triplet is error on pets, hazard, and vegetables groups, respectively.

Model	adv (%)	flip (%)
Resnet V2 50	87, 85, 57	1.3, 0.0, 0.0
Inception V3**	89, 87, 74	1.5, 0.5, 0.0

B Details of retinal blurring layer

B.1 Computing the primate eccentricity map

Let d_{viewer} be the distance (in meters) of the viewer from the display and d_{hw} be the height and width of a square image (in meters). For every spatial position (in meters) $c = (x, y) \in R^2$ in the image we compute the retinal eccentricity (in radians) as follows:

$$\theta\left(c\right) = \tan^{-1}(\frac{||c||_2}{d_{viewer}}) \tag{1}$$

and turn this into a target resolution in units of radians

$$r_{rad}\left(c\right) = \min\left(\alpha\theta\left(c\right), \beta\right). \tag{2}$$

We then turn this target resolution into a target spatial resolution in the plane of the screen,

$$r_m\left(c\right) = r_{rad}\left(c\right)\left(1 + \tan^2\left(\theta(c)\right)\right), \tag{3}$$

$$r_{pixel}\left(c\right) = r_m\left(c\right) \cdot \text{[pixels per meter]}. \tag{4}$$

This spatial resolution for two point discrimination is then converted into a corresponding low-pass cutoff frequency, in units of cycles per pixel,

$$f\left(c\right) = \frac{\pi}{r_{pixel}}, \tag{5}$$

where the numerator is π rather than 2π since the two point discrimination distance r_{pixel} is half the wavelength.

Finally, this target low-pass spatial frequency $f\left(c\right)$ for each pixel is used to linearly interpolate each pixel value from the corresponding pixel in a set of low pass filtered images, as described in the following algorithm (all operations on matrices are assumed to be performed elementwise), We additionally cropped $X_{retinal}$ to 90% width before use, to remove artifacts from the image edge.

Note that because the per-pixel blurring is performed using linear interpolation into images that were low-pass filtered in Fourier space, this transformation is both fast to compute and fully differentiable.

C Calculating probability of coarse class

To calculate the probability a model assigns to a coarse class, we summed probabilities assigned to the individual classes within the coarse class. Let S_{target} be the set of all individual labels

Algorithm 1 Applying retinal blur to an image

$X_{img} \leftarrow$ input image
$F \leftarrow$ image containing corresponding target lowpass frequency for each pixel, computed from $f(c)$
$\tilde{X} \leftarrow \text{FFT}(X_{img})$
$G \leftarrow$ norm of spatial frequency at each position in Y
CUTOFF_FREQS $\leftarrow$ list of frequencies to use as cutoffs for low-pass filtering
for f' in CUTOFF_FREQS **do**
    $\tilde{Y}_{f'} \leftarrow \tilde{X} \odot \exp\left(-\frac{G^2}{f'^2}\right)$
    $Y_f \leftarrow \text{InverseFFT}(\tilde{Y}_{f'})$
end for
$w(c) \leftarrow$ linear interpolation coefficients for $F(c)$ into CUTOFF_FREQS $\quad \forall c$
$X_{retinal}(c) \leftarrow \sum_{f'} w_{f'}(c) Y_{f'}(c) \quad \forall c$

in the target coarse class. Let S_{other} be all other individual labels not in the target coarse class. $|S_{\text{target}}| + |S_{\text{other}}| = 1000$, since there are 1000 labels in ImageNet. Let Y be the coarse class variable and y_{target} be our target coarse class. We can compute the probability a model k assigns to a coarse class given image X as

$$P_k(Y = y_{\text{target}}|X) = \sum_{i \in S_{\text{target}}} P_k(Y = y_i|X) = \sigma\left(\log \frac{\sum_{i \in S_{\text{target}}} \tilde{F}_k\left(i|X\right)}{\sum_{i \in S_{\text{other}}} \tilde{F}_k\left(i|X\right)}\right) \tag{6}$$

where $\tilde{F}_k\left(i|X\right)$ is the unnormalized probability assigned to fine class i (in practice = exp($logits$) of class i). The coarse logit of the model with respect to the target class y_{target} is then $F_k(Y = y_{\text{target}}|X) = \log \frac{\sum_{i \in S_{\text{target}}} \tilde{F}_k(i|X)}{\sum_{i \in S_{\text{other}}} \tilde{F}_k(i|X)}$.

D Adversarial images generation.

In the pipeline, an image is drawn from the source coarse class and perturbed to be classified as an image from the target coarse class. The attack method we use, the iterative targeted attack [24], is performed as

$$\begin{aligned} \tilde{X}^n_{adv} &= X^{n-1}_{adv} - \alpha * \text{sign}(\nabla_{X^n}(J(X^n|y_{\text{target}}))), \\ X^n_{adv} &= \text{clip}\left(\tilde{X}^n_{adv}, [X - \epsilon, X + \epsilon]\right), \end{aligned} \tag{7}$$

where J is the cost function as described below, y_{target} is the label of the target class, α is the step size, $X^0_{adv} = X$ is the original clean image, and $X_{adv} = X^N_{adv}$ is the final adversarial image. We set $\alpha = 2$, and ϵ is given per-condition in Section 3.2.2. After optimization, any perturbation whose ℓ_∞-norm was less than ϵ was scaled to have ℓ_∞-norm of ϵ, for consistency across all perturbations.

Our goal was to create adversarial examples that transferred across many ML models before assessing their transferability to humans. To accomplish this, we created an ensemble from the geometric mean of several image classifiers, and performed the iterative attack on the ensemble loss [26]

$$J\left(X|y_{target}\right) = -\log\left[P_{\text{ens}}\left(y_{\text{target}}|X\right)\right], \tag{8}$$

$$P_{\text{ens}}\left(y|X\right) \propto \exp\left(\mathbb{E}_k[\log P_k\left(y|X\right)]\right), \tag{9}$$

where $P_k\left(y|X\right)$ is the coarse class probabilities from model k, and $P_{\text{ens}}\left(y|X\right)$ is the probability from the ensemble. In practice, $J\left(X|y_{target}\right)$ is equivalent to standard cross entropy loss based on coarse logits averaged across models in the ensemble (see Appendix C for the coarse logit definition).

To encourage a high transfer rate, we retained only adversarial examples that were successful against all 10 models for the `adv` condition and at least $7/10$ models for the `false` condition (see Section 3.2.2 for condition definitions).

E Convolutional Neural Network Models

Some of the models in our ensemble are from a publicly available pretrained checkpoints[3], and others are our own instances of the architectures, specifically trained for this experiment on ImageNet with the retinal layer prepended. To encourage invariance to image intensity scaling, we augmented each training batch with another batch with the same images but rescaled in the range of $[40, 255 - 40]$, instead of $[0, 255]$. Supplementary Table Supp.2 identifies all ten models used in the ensemble, and shows their top-1 accuracies, along with two holdout models that we used for evaluation.

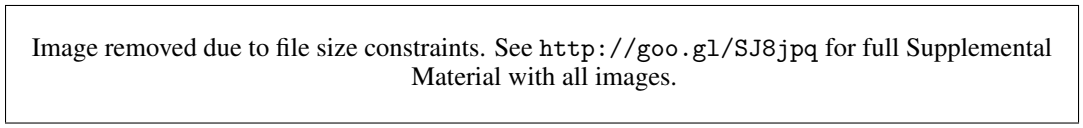


Figure Supp.2: **Adversarial Examples for false condition** (a) pets group. (b) hazard group. (c) vegetables group.

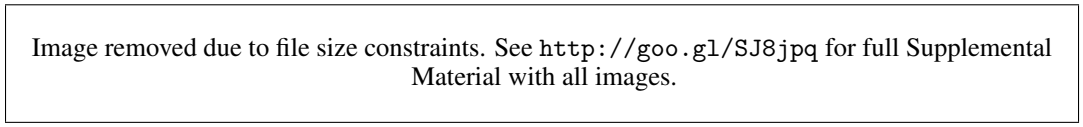


Figure Supp.3: **Adversarial Examples** pets group

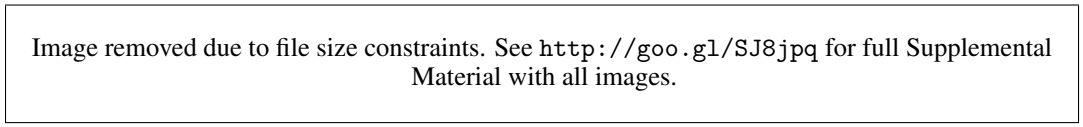


Figure Supp.4: **Adversarial Examples** hazard group

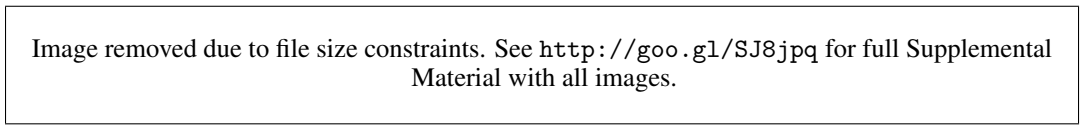


Figure Supp.5: **Adversarial Examples** vegetables group

F Image List from Imagenet

The specific imagenet images used from each class in the experiments in this paper are as follows:

dog:

'n02106382_564.JPEG',	'n02110958_598.JPEG',	'n02101556_13462.JPEG',
'n02113624_7358.JPEG',	'n02113799_2538.JPEG',	'n02091635_11576.JPEG',
'n02106382_2781.JPEG',	'n02112706_105.JPEG',	'n02095570_10951.JPEG',
'n02093859_5274.JPEG',	'n02109525_10825.JPEG',	'n02096294_1400.JPEG',
'n02086646_241.JPEG',	'n02098286_5642.JPEG',	'n02106382_9015.JPEG',
'n02090379_9754.JPEG',	'n02102318_10390.JPEG',	'n02086646_4202.JPEG',
'n02086910_5053.JPEG',	'n02113978_3051.JPEG',	'n02093859_3809.JPEG',
'n02105251_2485.JPEG',	'n02109525_35418.JPEG',	'n02108915_7834.JPEG',
'n02113624_430.JPEG',	'n02093256_7467.JPEG',	'n02087046_2701.JPEG',
'n02090379_8849.JPEG',	'n02093754_717.JPEG',	'n02086079_15905.JPEG',
'n02102480_4466.JPEG',	'n02107683_5333.JPEG',	'n02102318_8228.JPEG',
'n02099712_867.JPEG',	'n02094258_1958.JPEG',	'n02109047_25075.JPEG',
'n02113624_4304.JPEG',	'n02097474_10985.JPEG',	'n02091032_3832.JPEG',
'n02085620_859.JPEG',	'n02110806_582.JPEG',	'n02085782_8327.JPEG',
'n02094258_5318.JPEG',	'n02087046_5721.JPEG',	'n02095570_746.JPEG',
'n02099601_3771.JPEG',	'n02102480_41.JPEG',	'n02086910_1048.JPEG',

[3]https://github.com/tensorflow/models/tree/master/research/slim

’n02094114_7299.JPEG’, ’n02108551_13160.JPEG’, ’n02110185_9847.JPEG’,
’n02097298_13025.JPEG’, ’n02097298_16751.JPEG’, ’n02091467_555.JPEG’,
’n02113799_2504.JPEG’, ’n02085782_14116.JPEG’, ’n02097474_13885.JPEG’,
’n02105251_8108.JPEG’, ’n02113799_3415.JPEG’, ’n02095570_8170.JPEG’,
’n02088238_1543.JPEG’, ’n02097047_6.JPEG’, ’n02104029_5268.JPEG’,
’n02100583_11473.JPEG’, ’n02113978_6888.JPEG’, ’n02104365_1737.JPEG’,
’n02096177_4779.JPEG’, ’n02107683_5303.JPEG’, ’n02108915_11155.JPEG’,
’n02086910_1872.JPEG’, ’n02106550_8383.JPEG’, ’n02088094_2191.JPEG’,
’n02085620_11897.JPEG’, ’n02096051_4802.JPEG’, ’n02100735_3641.JPEG’,
’n02091032_1389.JPEG’, ’n02106382_4671.JPEG’, ’n02097298_9059.JPEG’,
’n02107312_280.JPEG’, ’n02111889_86.JPEG’, ’n02113978_5397.JPEG’,
’n02097209_3461.JPEG’, ’n02089867_1115.JPEG’, ’n02097658_4987.JPEG’,
’n02094114_4125.JPEG’, ’n02100583_130.JPEG’, ’n02112137_5859.JPEG’,
’n02113799_19636.JPEG’, ’n02088094_5488.JPEG’, ’n02089078_393.JPEG’,
’n02098413_1794.JPEG’, ’n02113799_1970.JPEG’, ’n02091032_3655.JPEG’,
’n02105855_11127.JPEG’, ’n02096294_3025.JPEG’, ’n02094114_4831.JPEG’,
’n02111889_10472.JPEG’, ’n02113624_9125.JPEG’, ’n02097474_9719.JPEG’,
’n02094433_2451.JPEG’, ’n02095889_6464.JPEG’, ’n02093256_458.JPEG’,
’n02091134_2732.JPEG’, ’n02091244_2622.JPEG’, ’n02094114_2169.JPEG’,
’n02090622_2337.JPEG’, ’n02101556_6764.JPEG’, ’n02096051_1459.JPEG’,
’n02087046_9056.JPEG’, ’n02098105_8405.JPEG’, ’n02112137_5696.JPEG’,
’n02110806_7949.JPEG’, ’n02097298_2420.JPEG’, ’n02085620_6814.JPEG’,
’n02108915_1703.JPEG’, ’n02100877_19273.JPEG’, ’n02106550_3765.JPEG’,
’n02107312_3524.JPEG’, ’n02111889_2963.JPEG’, ’n02113624_9129.JPEG’,
’n02097047_3200.JPEG’, ’n02093256_8365.JPEG’, ’n02093991_9420.JPEG’,
’n02112137_1635.JPEG’, ’n02111129_3530.JPEG’, ’n02101006_8123.JPEG’,
’n02102040_5033.JPEG’, ’n02113624_437.JPEG’, ’n02090622_5866.JPEG’,
’n02110806_3711.JPEG’, ’n02112137_14788.JPEG’, ’n02105162_7406.JPEG’,
’n02097047_5061.JPEG’, ’n02108422_11587.JPEG’, ’n02091467_4265.JPEG’,
’n02091467_12683.JPEG’, ’n02104365_3628.JPEG’, ’n02086646_3314.JPEG’,
’n02099849_736.JPEG’, ’n02100735_8112.JPEG’, ’n02112018_12764.JPEG’,
’n02093428_11175.JPEG’, ’n02110627_9822.JPEG’, ’n02107142_24318.JPEG’,
’n02105162_5489.JPEG’, ’n02093754_5904.JPEG’, ’n02110958_215.JPEG’,
’n02095314_4027.JPEG’, ’n02109961_3250.JPEG’, ’n02108551_7343.JPEG’,
’n02110627_10272.JPEG’, ’n02088364_3099.JPEG’, ’n02110806_2721.JPEG’,
’n02095314_2261.JPEG’, ’n02106550_9870.JPEG’, ’n02107574_3991.JPEG’,
’n02095570_3288.JPEG’, ’n02086079_39042.JPEG’, ’n02096294_9416.JPEG’,
’n02110806_6528.JPEG’, ’n02088466_11397.JPEG’, ’n02092002_996.JPEG’,
’n02098413_8605.JPEG’, ’n02085620_712.JPEG’, ’n02100236_3011.JPEG’,
’n02086646_7788.JPEG’, ’n02085620_4661.JPEG’, ’n02098105_1746.JPEG’,
’n02113624_8608.JPEG’, ’n02097474_1168.JPEG’, ’n02107683_1496.JPEG’,
’n02110185_12849.JPEG’, ’n02085620_11946.JPEG’, ’n02087394_16385.JPEG’,
’n02110806_22671.JPEG’, ’n02113624_526.JPEG’, ’n02096294_12642.JPEG’,
’n02113023_7510.JPEG’, ’n02088364_13285.JPEG’, ’n02095889_2977.JPEG’,
’n02105056_9215.JPEG’, ’n02102318_9744.JPEG’, ’n02097298_11834.JPEG’,
’n02111277_16201.JPEG’, ’n02085782_8518.JPEG’, ’n02113978_11280.JPEG’,
’n02106382_10700.JPEG’.

cat:

’n02123394_661.JPEG’, ’n02123045_11954.JPEG’, ’n02123394_3695.JPEG’,
’n02123394_2692.JPEG’, ’n02123597_12166.JPEG’, ’n02123045_7014.JPEG’,
’n02123159_2777.JPEG’, ’n02123394_684.JPEG’, ’n02124075_543.JPEG’,
’n02123597_7557.JPEG’, ’n02124075_7857.JPEG’, ’n02123597_3770.JPEG’,
’n02124075_4986.JPEG’, ’n02123045_568.JPEG’, ’n02123394_1541.JPEG’,
’n02123597_3498.JPEG’, ’n02123597_10304.JPEG’, ’n02123394_2084.JPEG’,
’n02123597_5283.JPEG’, ’n02123597_13807.JPEG’, ’n02124075_12282.JPEG’,
’n02123597_8575.JPEG’, ’n02123045_11787.JPEG’, ’n02123394_888.JPEG’,
’n02123045_1815.JPEG’, ’n02123394_7614.JPEG’, ’n02123597_27865.JPEG’,
’n02124075_1279.JPEG’, ’n02123394_4775.JPEG’, ’n02123394_976.JPEG’,

’n02123394_8385.JPEG’, ’n02123597_14791.JPEG’, ’n02123045_10424.JPEG’,
’n02123597_7698.JPEG’, ’n02124075_8140.JPEG’, ’n02123045_3754.JPEG’,
’n02123597_1819.JPEG’, ’n02123597_395.JPEG’, ’n02123394_415.JPEG’,
’n02124075_9747.JPEG’, ’n02123045_9467.JPEG’, ’n02123159_6842.JPEG’,
’n02123394_9611.JPEG’, ’n02123597_7283.JPEG’, ’n02123597_11799.JPEG’,
’n02123597_660.JPEG’, ’n02123045_7511.JPEG’, ’n02123597_10723.JPEG’,
’n02123159_7836.JPEG’, ’n02123597_14530.JPEG’, ’n02123597_28555.JPEG’,
’n02123394_6079.JPEG’, ’n02123394_6792.JPEG’, ’n02123597_11564.JPEG’,
’n02123597_8916.JPEG’, ’n02124075_123.JPEG’, ’n02123045_5150.JPEG’,
’n02124075_353.JPEG’, ’n02123597_12941.JPEG’, ’n02123045_10095.JPEG’,
’n02123597_6533.JPEG’, ’n02123045_4611.JPEG’, ’n02123597_754.JPEG’,
’n02123394_8561.JPEG’, ’n02123597_6409.JPEG’, ’n02123159_4909.JPEG’,
’n02123597_564.JPEG’, ’n02123394_1633.JPEG’, ’n02123394_1196.JPEG’,
’n02123394_2787.JPEG’, ’n02124075_10542.JPEG’, ’n02123597_6242.JPEG’,
’n02123597_3063.JPEG’, ’n02123597_13164.JPEG’, ’n02123045_7449.JPEG’,
’n02123045_13299.JPEG’, ’n02123394_8165.JPEG’, ’n02123394_1852.JPEG’,
’n02123597_8771.JPEG’, ’n02123159_6581.JPEG’, ’n02123394_5906.JPEG’,
’n02124075_2747.JPEG’, ’n02124075_11383.JPEG’, ’n02123597_3919.JPEG’,
’n02123394_2514.JPEG’, ’n02124075_7423.JPEG’, ’n02123394_6968.JPEG’,
’n02123045_4850.JPEG’, ’n02123045_10689.JPEG’, ’n02124075_13539.JPEG’,
’n02123597_13378.JPEG’, ’n02123159_4847.JPEG’, ’n02123394_1798.JPEG’,
’n02123597_27951.JPEG’, ’n02123159_587.JPEG’, ’n02123597_1825.JPEG’,
’n02123159_2200.JPEG’, ’n02123597_12.JPEG’, ’n02123597_6778.JPEG’,
’n02123597_6693.JPEG’, ’n02123045_11782.JPEG’, ’n02123597_13706.JPEG’,
’n02123394_9032.JPEG’, ’n02124075_4459.JPEG’, ’n02123597_13752.JPEG’,
’n02123394_2285.JPEG’, ’n02123597_1410.JPEG’, ’n02123159_6134.JPEG’,
’n02123597_11290.JPEG’, ’n02123597_6347.JPEG’, ’n02123394_1789.JPEG’,
’n02123045_11255.JPEG’, ’n02123394_6096.JPEG’, ’n02123394_4081.JPEG’,
’n02123394_5679.JPEG’, ’n02123394_2471.JPEG’, ’n02123159_5797.JPEG’,
’n02123597_13894.JPEG’, ’n02124075_10854.JPEG’, ’n02123394_8605.JPEG’,
’n02124075_8281.JPEG’, ’n02123597_11724.JPEG’, ’n02123394_8242.JPEG’,
’n02123394_3569.JPEG’, ’n02123597_10639.JPEG’, ’n02123045_3818.JPEG’,
’n02124075_6459.JPEG’, ’n02123394_185.JPEG’, ’n02123597_8961.JPEG’,
’n02124075_9743.JPEG’, ’n02123394_1627.JPEG’, ’n02123597_13175.JPEG’,
’n02123045_2694.JPEG’, ’n02123597_4537.JPEG’, ’n02123597_6400.JPEG’,
’n02123045_7423.JPEG’, ’n02123597_3004.JPEG’, ’n02123394_2988.JPEG’,
’n02124075_9512.JPEG’, ’n02123394_6318.JPEG’, ’n02123597_1843.JPEG’,
’n02124075_2053.JPEG’, ’n02123597_3828.JPEG’, ’n02123394_14.JPEG’,
’n02123394_8141.JPEG’, ’n02124075_1624.JPEG’, ’n02123597_459.JPEG’,
’n02124075_6405.JPEG’, ’n02123045_8595.JPEG’, ’n02123159_3226.JPEG’,
’n02124075_9141.JPEG’, ’n02123597_2031.JPEG’, ’n02123045_2354.JPEG’,
’n02123597_6710.JPEG’, ’n02123597_6613.JPEG’, ’n02123159_1895.JPEG’,
’n02123394_2953.JPEG’, ’n02123394_5846.JPEG’, ’n02123394_513.JPEG’,
’n02123045_16637.JPEG’, ’n02123394_7848.JPEG’, ’n02123394_3229.JPEG’,
’n02123045_8881.JPEG’, ’n02123394_8250.JPEG’, ’n02124075_7651.JPEG’,
’n02123394_200.JPEG’, ’n02123394_2814.JPEG’, ’n02123045_6445.JPEG’,
’n02123394_2467.JPEG’, ’n02123045_3317.JPEG’, ’n02123597_1422.JPEG’,
’n02123597_13442.JPEG’, ’n02123394_8225.JPEG’, ’n02123597_9337.JPEG’,
’n02123394_32.JPEG’, ’n02123394_2193.JPEG’, ’n02123394_1625.JPEG’,
’n02123597_8799.JPEG’, ’n02123597_13241.JPEG’, ’n02123597_7681.JPEG’,
’n02123597_4550.JPEG’, ’n02123597_3896.JPEG’, ’n02123394_9554.JPEG’,
’n02124075_13600.JPEG’, ’n02123394_571.JPEG’, ’n02123597_10886.JPEG’,
’n02123045_6741.JPEG’, ’n02123045_10438.JPEG’, ’n02123045_9954.JPEG’.

spider:

’n01775062_517.JPEG’, ’n01774750_18017.JPEG’, ’n01774384_13186.JPEG’,
’n01774750_3115.JPEG’, ’n01775062_5075.JPEG’, ’n01773549_1541.JPEG’,
’n01775062_4867.JPEG’, ’n01775062_8156.JPEG’, ’n01774750_7128.JPEG’,
’n01775062_4632.JPEG’, ’n01773549_8734.JPEG’, ’n01773549_2274.JPEG’,

’n01773549_10298.JPEG’, ’n01774384_1811.JPEG’, ’n01774750_7498.JPEG’,
’n01774750_10265.JPEG’, ’n01773549_1964.JPEG’, ’n01774750_3268.JPEG’,
’n01773549_6095.JPEG’, ’n01775062_8812.JPEG’, ’n01774750_10919.JPEG’,
’n01775062_1180.JPEG’, ’n01773549_7275.JPEG’, ’n01773549_9346.JPEG’,
’n01773549_8243.JPEG’, ’n01775062_3127.JPEG’, ’n01773549_10608.JPEG’,
’n01773549_3442.JPEG’, ’n01773157_1487.JPEG’, ’n01774750_7775.JPEG’,
’n01775062_419.JPEG’, ’n01774750_7638.JPEG’, ’n01775062_847.JPEG’,
’n01774750_3154.JPEG’, ’n01773549_1534.JPEG’, ’n01773157_1039.JPEG’,
’n01775062_5644.JPEG’, ’n01775062_8525.JPEG’, ’n01773797_216.JPEG’,
’n01775062_900.JPEG’, ’n01774750_8513.JPEG’, ’n01774750_3424.JPEG’,
’n01774750_3085.JPEG’, ’n01775062_3662.JPEG’, ’n01774384_15681.JPEG’,
’n01774750_326.JPEG’, ’n01773157_9503.JPEG’, ’n01774750_3332.JPEG’,
’n01774750_2799.JPEG’, ’n01773157_10606.JPEG’, ’n01773157_1905.JPEG’,
’n01773549_379.JPEG’, ’n01773797_597.JPEG’, ’n01773157_3226.JPEG’,
’n01774750_7875.JPEG’, ’n01774384_16102.JPEG’, ’n01773549_2832.JPEG’,
’n01775062_5072.JPEG’, ’n01773549_4278.JPEG’, ’n01773549_5854.JPEG’,
’n01774384_1998.JPEG’, ’n01774750_13875.JPEG’, ’n01775062_8270.JPEG’,
’n01773549_2941.JPEG’, ’n01774750_5235.JPEG’, ’n01773549_4150.JPEG’,
’n01774750_6217.JPEG’, ’n01775062_3137.JPEG’, ’n01774750_5480.JPEG’,
’n01774384_11955.JPEG’, ’n01775062_8376.JPEG’, ’n01773157_2688.JPEG’,
’n01773549_6825.JPEG’, ’n01774750_10422.JPEG’, ’n01774384_20786.JPEG’,
’n01773549_398.JPEG’, ’n01773549_4965.JPEG’, ’n01774750_7470.JPEG’,
’n01775062_1379.JPEG’, ’n01774384_2399.JPEG’, ’n01773549_9799.JPEG’,
’n01775062_305.JPEG’, ’n01774384_15519.JPEG’, ’n01774750_3333.JPEG’,
’n01774750_2604.JPEG’, ’n01774750_3134.JPEG’, ’n01774750_4646.JPEG’,
’n01775062_5009.JPEG’, ’n01774750_10200.JPEG’, ’n01775062_7964.JPEG’,
’n01774384_2458.JPEG’, ’n01773797_3333.JPEG’, ’n01774750_9987.JPEG’,
’n01773549_5790.JPEG’, ’n01773549_854.JPEG’, ’n01774750_11370.JPEG’,
’n01774750_10698.JPEG’, ’n01774750_9287.JPEG’, ’n01773797_6703.JPEG’,
’n01773797_931.JPEG’, ’n01773549_5280.JPEG’, ’n01773797_5385.JPEG’,
’n01773797_1098.JPEG’, ’n01774750_436.JPEG’, ’n01774384_13770.JPEG’,
’n01774750_9780.JPEG’, ’n01774750_8640.JPEG’, ’n01774750_653.JPEG’,
’n01774384_12554.JPEG’, ’n01774750_9716.JPEG’

snake:

’n01737021_7081.JPEG’, ’n01728572_16119.JPEG’, ’n01735189_10620.JPEG’,
’n01751748_3573.JPEG’, ’n01729322_6690.JPEG’, ’n01735189_20703.JPEG’,
’n01734418_4792.JPEG’, ’n01749939_2784.JPEG’, ’n01729977_4113.JPEG’,
’n01756291_6505.JPEG’, ’n01742172_3003.JPEG’, ’n01728572_19317.JPEG’,
’n01739381_5838.JPEG’, ’n01737021_1381.JPEG’, ’n01749939_4704.JPEG’,
’n01755581_10792.JPEG’, ’n01729977_9474.JPEG’, ’n01744401_11909.JPEG’,
’n01739381_10303.JPEG’, ’n01749939_820.JPEG’, ’n01728572_27743.JPEG’,
’n01734418_12057.JPEG’, ’n01742172_8636.JPEG’, ’n01729977_14112.JPEG’,
’n01739381_6286.JPEG’, ’n01734418_761.JPEG’, ’n01740131_13437.JPEG’,
’n01728920_9571.JPEG’, ’n01753488_4234.JPEG’, ’n01749939_5712.JPEG’,
’n01739381_6072.JPEG’, ’n01739381_7683.JPEG’, ’n01729322_9202.JPEG’,
’n01751748_13413.JPEG’, ’n01756291_4626.JPEG’, ’n01742172_9733.JPEG’,
’n01737021_12610.JPEG’, ’n01739381_87.JPEG’, ’n01729977_1134.JPEG’,
’n01753488_637.JPEG’, ’n01748264_18478.JPEG’, ’n01728572_22360.JPEG’,
’n01737021_3386.JPEG’, ’n01751748_560.JPEG’, ’n01751748_18223.JPEG’,
’n01749939_5750.JPEG’, ’n01748264_7044.JPEG’, ’n01739381_1163.JPEG’,
’n01751748_311.JPEG’, ’n01756291_9028.JPEG’, ’n01739381_10473.JPEG’,
’n01728572_1415.JPEG’, ’n01729322_10918.JPEG’, ’n01748264_653.JPEG’,
’n01753488_10957.JPEG’, ’n01756291_3990.JPEG’, ’n01756291_11915.JPEG’,
’n01756291_6776.JPEG’, ’n01740131_11661.JPEG’, ’n01729977_5715.JPEG’,
’n01737021_16733.JPEG’, ’n01753488_15197.JPEG’, ’n01744401_7248.JPEG’,
’n01728572_7661.JPEG’, ’n01740131_13680.JPEG’, ’n01729322_5446.JPEG’,
’n01749939_6508.JPEG’, ’n01748264_2140.JPEG’, ’n01729977_16782.JPEG’,
’n01748264_7602.JPEG’, ’n01756291_17857.JPEG’, ’n01729977_461.JPEG’,

’n01742172_20552.JPEG’, ’n01735189_3258.JPEG’, ’n01728920_9265.JPEG’,
’n01748264_18133.JPEG’, ’n01748264_16699.JPEG’, ’n01739381_1006.JPEG’,
’n01753488_10555.JPEG’, ’n01751748_3202.JPEG’, ’n01734418_3929.JPEG’,
’n01751748_5908.JPEG’, ’n01751748_8470.JPEG’, ’n01739381_3598.JPEG’,
’n01739381_255.JPEG’, ’n01729977_15657.JPEG’, ’n01748264_21477.JPEG’,
’n01751748_2912.JPEG’, ’n01728920_9154.JPEG’, ’n01728572_17552.JPEG’,
’n01740131_14560.JPEG’, ’n01729322_5947.JPEG’.

Broccoli:

’n07714990_8640.JPEG’, ’n07714990_5643.JPEG’, ’n07714990_7777.JPEG’,
’n07714990_888.JPEG’, ’n07714990_3398.JPEG’, ’n07714990_4576.JPEG’,
’n07714990_8554.JPEG’, ’n07714990_1957.JPEG’, ’n07714990_4201.JPEG’,
’n07714990_3130.JPEG’, ’n07714990_4115.JPEG’, ’n07714990_524.JPEG’,
’n07714990_6504.JPEG’, ’n07714990_3125.JPEG’, ’n07714990_5838.JPEG’,
’n07714990_1779.JPEG’, ’n07714990_6393.JPEG’, ’n07714990_1409.JPEG’,
’n07714990_4962.JPEG’, ’n07714990_7282.JPEG’, ’n07714990_7314.JPEG’,
’n07714990_11933.JPEG’, ’n07714990_1202.JPEG’, ’n07714990_3626.JPEG’,
’n07714990_7873.JPEG’, ’n07714990_3325.JPEG’, ’n07714990_3635.JPEG’,
’n07714990_12524.JPEG’, ’n07714990_14952.JPEG’, ’n07714990_7048.JPEG’,
’n07714990_500.JPEG’, ’n07714990_7950.JPEG’, ’n07714990_2445.JPEG’,
’n07714990_1294.JPEG’, ’n07714990_7336.JPEG’, ’n07714990_14743.JPEG’,
’n07714990_1423.JPEG’, ’n07714990_2185.JPEG’, ’n07714990_6566.JPEG’,
’n07714990_567.JPEG’, ’n07714990_1532.JPEG’, ’n07714990_5212.JPEG’,
’n07714990_8971.JPEG’, ’n07714990_6116.JPEG’, ’n07714990_5462.JPEG’,
’n07714990_7644.JPEG’, ’n07714990_8596.JPEG’, ’n07714990_1138.JPEG’,
’n07714990_15078.JPEG’, ’n07714990_1602.JPEG’, ’n07714990_2460.JPEG’,
’n07714990_159.JPEG’, ’n07714990_9445.JPEG’, ’n07714990_471.JPEG’,
’n07714990_1777.JPEG’, ’n07714990_9760.JPEG’, ’n07714990_1528.JPEG’,
’n07714990_12338.JPEG’, ’n07714990_2201.JPEG’, ’n07714990_6850.JPEG’,
’n07714990_4492.JPEG’, ’n07714990_7791.JPEG’, ’n07714990_9752.JPEG’,
’n07714990_1702.JPEG’, ’n07714990_3682.JPEG’, ’n07714990_14342.JPEG’,
’n07714990_2661.JPEG’, ’n07714990_5467.JPEG’.

Cabbage:

’n07714571_14784.JPEG’, ’n07714571_4795.JPEG’, ’n07714571_11969.JPEG’,
’n07714571_1394.JPEG’, ’n07714571_4155.JPEG’, ’n07714571_3624.JPEG’,
’n07714571_13753.JPEG’, ’n07714571_7351.JPEG’, ’n07714571_10316.JPEG’,
’n07714571_7235.JPEG’, ’n07714571_17716.JPEG’, ’n07714571_1639.JPEG’,
’n07714571_5107.JPEG’, ’n07714571_4109.JPEG’, ’n07714571_11878.JPEG’,
’n07714571_15910.JPEG’, ’n07714571_14401.JPEG’, ’n07714571_2741.JPEG’,
’n07714571_8576.JPEG’, ’n07714571_1624.JPEG’, ’n07714571_13479.JPEG’,
’n07714571_2715.JPEG’, ’n07714571_3676.JPEG’, ’n07714571_12371.JPEG’,
’n07714571_4829.JPEG’, ’n07714571_3922.JPEG’, ’n07714571_10377.JPEG’,
’n07714571_8040.JPEG’, ’n07714571_8147.JPEG’, ’n07714571_10377.JPEG’,
’n07714571_8040.JPEG’, ’n07714571_5730.JPEG’, ’n07714571_16460.JPEG’,
’n07714571_8198.JPEG’, ’n07714571_1095.JPEG’, ’n07714571_3922.JPEG’,
’n07714571_7745.JPEG’, ’n07714571_6301.JPEG’.

Group Normalization

Yuxin Wu Kaiming He

Facebook AI Research (FAIR)

{yuxinwu,kaiminghe}@fb.com

Abstract

Batch Normalization (BN) is a milestone technique in the development of deep learning, enabling various networks to train. However, normalizing along the batch dimension introduces problems — BN's error increases rapidly when the batch size becomes smaller, caused by inaccurate batch statistics estimation. This limits BN's usage for training larger models and transferring features to computer vision tasks including detection, segmentation, and video, which require small batches constrained by memory consumption. In this paper, we present Group Normalization (GN) as a simple alternative to BN. GN divides the channels into groups and computes within each group the mean and variance for normalization. GN's computation is independent of batch sizes, and its accuracy is stable in a wide range of batch sizes. On ResNet-50 trained in ImageNet, GN has 10.6% lower error than its BN counterpart when using a batch size of 2; when using typical batch sizes, GN is comparably good with BN and outperforms other normalization variants. Moreover, GN can be naturally transferred from pre-training to fine-tuning. GN can outperform its BN-based counterparts for object detection and segmentation in COCO,[1] *and for video classification in Kinetics, showing that GN can effectively replace the powerful BN in a variety of tasks. GN can be easily implemented by a few lines of code in modern libraries.*

1. Introduction

Batch Normalization (Batch Norm or BN) [26] has been established as a very effective component in deep learning, largely helping push the frontier in computer vision [59, 20] and beyond [54]. BN normalizes the features by the mean and variance computed within a (mini-)batch. This has been shown by many practices to ease optimization and enable very deep networks to converge. The stochastic uncertainty of the batch statistics also acts as a regularizer that can benefit generalization. BN has been a foundation of many state-of-the-art computer vision algorithms.

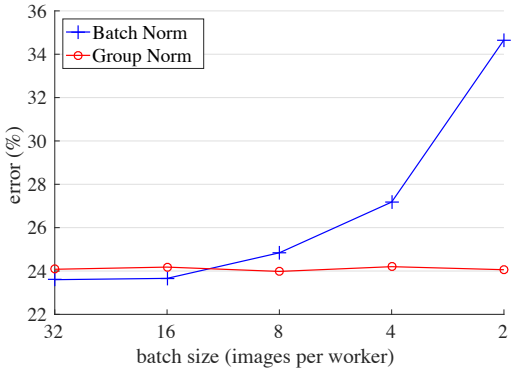


Figure 1. **ImageNet classification error *vs*. batch sizes**. This is a ResNet-50 model trained in the ImageNet training set using 8 workers (GPUs), evaluated in the validation set.

Despite its great success, BN exhibits drawbacks that are also caused by its distinct behavior of normalizing along the batch dimension. In particular, it is required for BN to work with a *sufficiently large batch size* (*e.g.*, 32 per worker[2] [26, 59, 20]). A small batch leads to inaccurate estimation of the batch statistics, and *reducing BN's batch size increases the model error dramatically* (Figure 1). As a result, many recent models [59, 20, 57, 24, 63] are trained with non-trivial batch sizes that are memory-consuming. The heavy reliance on BN's effectiveness to train models in turn prohibits people from exploring higher-capacity models that would be limited by memory.

The restriction on batch sizes is more demanding in computer vision tasks including detection [12, 47, 18], segmentation [38, 18], video recognition [60, 6], and other high-level systems built on them. For example, the Fast/er and Mask R-CNN frameworks [12, 47, 18] use a batch size of 1 or 2 images because of higher resolution, where BN is "frozen" by transforming to a linear layer [20]; in video classification with 3D convolutions [60, 6], the presence of spatial-temporal features introduces a trade-off between the temporal length and batch size. The usage of BN often requires these systems to compromise between the model design and batch sizes.

[1] https://github.com/facebookresearch/Detectron/blob/master/projects/GN.

[2] In the context of this paper, we use "batch size" to refer to the number of samples *per worker* (*e.g.*, GPU). BN's statistics are computed for each worker, but *not* broadcast across workers, as is standard in many libraries.

This paper presents Group Normalization (GN) as a simple alternative to BN. We notice that many classical features like SIFT [39] and HOG [9] are *group-wise* features and involve *group-wise normalization*. For example, a HOG vector is the outcome of several spatial cells where each cell is represented by a normalized orientation histogram. Analogously, we propose GN as a layer that divides channels into groups and normalizes the features within each group (Figure 2). GN does not exploit the batch dimension, and its computation is independent of batch sizes.

GN behaves very stably over a wide range of batch sizes (Figure 1). With a batch size of 2 samples, GN has 10.6% lower error than its BN counterpart for ResNet-50 [20] in ImageNet [50]. With a regular batch size, GN is comparably good as BN (with a gap of ~0.5%) and outperforms other normalization variants [3, 61, 51]. Moreover, although the batch size may change, GN can naturally transfer from pre-training to fine-tuning. GN shows improved results *vs*. its BN counterpart on Mask R-CNN for COCO object detection and segmentation [37], and on 3D convolutional networks for Kinetics video classification [30]. The effectiveness of GN in ImageNet, COCO, and Kinetics demonstrates that GN is a competitive alternative to BN that has been dominant in these tasks.

There have been existing methods, such as Layer Normalization (LN) [3] and Instance Normalization (IN) [61] (Figure 2), that also avoid normalizing along the batch dimension. These methods are effective for training sequential models (RNN/LSTM [49, 22]) or generative models (GANs [15, 27]). But as we will show by experiments, both LN and IN have limited success in visual recognition, for which GN presents better results. Conversely, GN could be used in place of LN and IN and thus is applicable for sequential or generative models. This is beyond the focus of this paper, but it is suggestive for future research.

2. Related Work

Normalization. It is well-known that normalizing the input data makes training faster [33]. To normalize hidden features, initialization methods [33, 14, 19] have been derived based on strong assumptions of feature distributions, which can become invalid when training evolves.

Normalization layers in deep networks had been widely used before the development of BN. Local Response Normalization (LRN) [40, 28, 32] was a component in AlexNet [32] and following models [64, 53, 58]. Unlike recent methods [26, 3, 61], LRN computes the statistics in a small neighborhood for each pixel.

Batch Normalization [26] performs more global normalization along the batch dimension (and as importantly, it suggests to do this for all layers). But the concept of "batch" is not always present, or it may change from time to time. For example, batch-wise normalization is not legitimate at inference time, so the mean and variance are pre-computed from the training set [26], often by running average; consequently, there is no normalization performed when testing. The pre-computed statistics may also change when the target data distribution changes [45]. These issues lead to inconsistency at training, transferring, and testing time. In addition, as aforementioned, reducing the batch size can have dramatic impact on the estimated batch statistics.

Several normalization methods [3, 61, 51, 2, 46] have been proposed to avoid exploiting the batch dimension. Layer Normalization (LN) [3] operates along the channel dimension, and Instance Normalization (IN) [61] performs BN-like computation but only for each sample (Figure 2). Instead of operating on features, Weight Normalization (WN) [51] proposes to normalize the filter weights. These methods do not suffer from the issues caused by the batch dimension, but they have not been able to approach BN's accuracy in many visual recognition tasks. We provide comparisons with these methods in context of the remaining sections.

Addressing small batches. Ioffe [25] proposes Batch Renormalization (BR) that alleviates BN's issue involving small batches. BR introduces two extra parameters that constrain the estimated mean and variance of BN within a certain range, reducing their drift when the batch size is small. BR has better accuracy than BN in the small-batch regime. But BR is also batch-dependent, and when the batch size decreases its accuracy still degrades [25].

There are also attempts to *avoid* using small batches. The object detector in [43] performs synchronized BN whose mean and variance are computed across multiple GPUs. However, this method does not solve the problem of small batches; instead, it migrates the algorithm problem to engineering and hardware demands, using a number of GPUs proportional to BN's requirements. Moreover, the synchronized BN computation prevents using *asynchronous* solvers (ASGD [10]), a practical solution to large-scale training widely used in industry. These issues can limit the scope of using synchronized BN.

Instead of addressing the batch statistics computation (*e.g*., [25, 43]), our normalization method inherently avoids this computation.

Group-wise computation. *Group convolutions* have been presented by AlexNet [32] for distributing a model into two GPUs. The concept of *groups* as a dimension for model design has been more widely studied recently. The work of ResNeXt [63] investigates the trade-off between depth, width, and groups, and it suggests that a larger number of groups can improve accuracy under similar computational cost. MobileNet [23] and Xception [7] exploit *channel-wise* (also called "depth-wise") convolutions, which are group convolutions with a group number equal to the channel

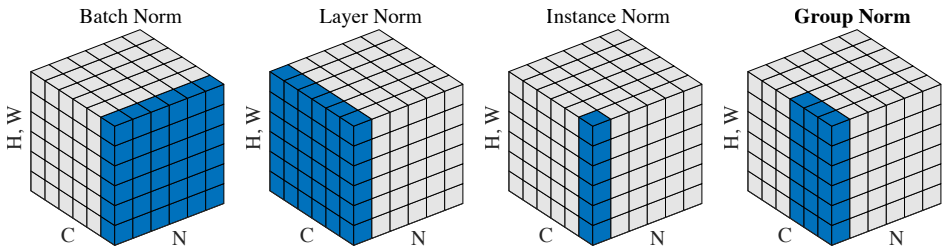


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

number. ShuffleNet [65] proposes a channel shuffle operation that permutes the axes of grouped features. These methods all involve dividing the channel dimension into groups. Despite the relation to these methods, GN does *not* require group convolutions. GN is a generic layer, as we evaluate in standard ResNets [20].

3. Group Normalization

The channels of visual representations are not entirely independent. Classical features of SIFT [39], HOG [9], and GIST [41] are *group-wise* representations by design, where each group of channels is constructed by some kind of histogram. These features are often processed by *group-wise normalization* over each histogram or each orientation. Higher-level features such as VLAD [29] and Fisher Vectors (FV) [44] are also group-wise features where a group can be thought of as the sub-vector computed with respect to a cluster.

Analogously, it is not necessary to think of deep neural network features as unstructured vectors. For example, for conv_1 (the first convolutional layer) of a network, it is reasonable to expect a filter and its horizontal flipping to exhibit similar distributions of filter responses on natural images. If conv_1 happens to approximately learn this pair of filters, or if the horizontal flipping (or other transformations) is made into the architectures by design [11, 8], then the corresponding channels of these filters can be normalized together.

The higher-level layers are more abstract and their behaviors are not as intuitive. However, in addition to orientations (SIFT [39], HOG [9], or [11, 8]), there are many factors that could lead to grouping, *e.g.*, frequency, shapes, illumination, textures. Their coefficients can be interdependent. In fact, a well-accepted computational model in neuroscience is to normalize across the cell responses [21, 52, 55, 5], "with various receptive-field centers (covering the visual field) and with various spatiotemporal frequency tunings" (p183, [21]); this can happen not only in the primary visual cortex, but also "throughout the visual system" [5]. Motivated by these works, we propose new generic group-wise normalization for deep neural networks.

3.1. Formulation

We first describe a general formulation of feature normalization, and then present GN in this formulation. A family of feature normalization methods, including BN, LN, IN, and GN, perform the following computation:

$$\hat{x}_i = \frac{1}{\sigma_i}(x_i - \mu_i). \tag{1}$$

Here x is the feature computed by a layer, and i is an index. In the case of 2D images, $i = (i_N, i_C, i_H, i_W)$ is a 4D vector indexing the features in (N, C, H, W) order, where N is the batch axis, C is the channel axis, and H and W are the spatial height and width axes.

μ and σ in (1) are the mean and standard deviation (std) computed by:

$$\mu_i = \frac{1}{m}\sum_{k\in\mathcal{S}_i} x_k, \quad \sigma_i = \sqrt{\frac{1}{m}\sum_{k\in\mathcal{S}_i}(x_k - \mu_i)^2 + \epsilon}, \tag{2}$$

with ϵ as a small constant. $\mathcal{S}_i$ is the set of pixels in which the mean and std are computed, and m is the size of this set. Many types of feature normalization methods mainly differ in how the set $\mathcal{S}_i$ is defined (Figure 2), discussed as follows.

In **Batch Norm** [26], the set $\mathcal{S}_i$ is defined as:

$$\mathcal{S}_i = \{k \mid k_C = i_C\}, \tag{3}$$

where i_C (and k_C) denotes the sub-index of i (and k) along the C axis. This means that the pixels sharing the same channel index are normalized together, *i.e.*, for each channel, BN computes μ and σ along the (N, H, W) axes. In **Layer Norm** [3], the set is:

$$\mathcal{S}_i = \{k \mid k_N = i_N\}, \tag{4}$$

meaning that LN computes μ and σ along the (C, H, W) axes for each sample. In **Instance Norm** [61], the set is:

$$\mathcal{S}_i = \{k \mid k_N = i_N, k_C = i_C\}. \tag{5}$$

meaning that IN computes μ and σ along the (H, W) axes for each sample and each channel. The relations among BN, LN, and IN are in Figure 2.

As in [26], all methods of BN, LN, and IN learn a per-channel linear transform to compensate for the possible lost of representational ability:

$$y_i = \gamma \hat{x}_i + \beta, \tag{6}$$

where γ and β are trainable scale and shift (indexed by i_C in all case, which we omit for simplifying notations).

Group Norm. Formally, a Group Norm layer computes μ and σ in a set $\mathcal{S}_i$ defined as:

$$\mathcal{S}_i = \{k \mid k_N = i_N, \lfloor \frac{k_C}{C/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor \}. \tag{7}$$

Here G is the number of groups, which is a pre-defined hyper-parameter ($G = 32$ by default). C/G is the number of channels per group. $\lfloor \cdot \rfloor$ is the floor operation, and "$\lfloor \frac{k_C}{C/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor$" means that the indexes i and k are in the same group of channels, assuming each group of channels are stored in a sequential order along the C axis. GN computes μ and σ along the (H, W) axes and along a group of $\frac{C}{G}$ channels. The computation of GN is illustrated in Figure 2 (rightmost), which is a simple case of 2 groups ($G = 2$) each having 3 channels.

Given $\mathcal{S}_i$ in Eqn.(7), a GN layer is defined by Eqn.(1), (2), and (6). Specifically, the pixels in the same group are normalized together by the same μ and σ. GN also learns the per-channel γ and β.

Relation to Prior Work. LN, IN, and GN all perform independent computations along the batch axis. The two extreme cases of GN are equivalent to LN and IN (Figure 2).

Relation to Layer Normalization [3]. GN becomes LN if we set the group number as $G = 1$. LN assumes *all* channels in a layer make "similar contributions" [3]. Unlike the case of fully-connected layers studied in [3], this assumption can be less valid with the presence of convolutions, as discussed in [3]. GN is less restricted than LN, because each group of channels (instead of all of them) are assumed to subject to the shared mean and variance; the model still has flexibility of learning a different distribution for each group. This leads to improved representational power of GN over LN, as shown by the lower training and validation error in experiments (Figure 4).

Relation to Instance Normalization [61]. GN becomes IN if we set the group number as $G = C$ (*i.e.*, one channel per group). But IN can only rely on the spatial dimension for computing the mean and variance and it misses the opportunity of exploiting the channel dependence.

3.2. Implementation

GN can be easily implemented by a few lines of code in PyTorch [42] and TensorFlow [1] where automatic differentiation is supported. Figure 3 shows the code based on TensorFlow. In fact, we only need to specify how the mean and variance ("moments") are computed, along the appropriate axes as defined by the normalization method.

```
def GroupNorm(x, gamma, beta, G, eps=1e−5):
    # x: input features with shape [N,C,H,W]
    # gamma, beta: scale and offset, with shape [1,C,1,1]
    # G: number of groups for GN

    N, C, H, W = x.shape
    x = tf.reshape(x, [N, G, C // G, H, W])

    mean, var = tf.nn.moments(x, [2, 3, 4], keep_dims=True)
    x = (x − mean) / tf.sqrt(var + eps)

    x = tf.reshape(x, [N, C, H, W])

    return x ∗ gamma + beta
```

Figure 3. Python code of Group Norm based on TensorFlow.

4. Experiments

4.1. Image Classification in ImageNet

We experiment in the ImageNet classification dataset [50] with 1000 classes. We train on the ~1.28M training images and evaluate on the 50,000 validation images, using the ResNet models [20].

Implementation details. As standard practice [20, 17], we use 8 GPUs to train all models, and the batch mean and variance of BN are computed *within* each GPU. We use the method of [19] to initialize all convolutions for all models. We use 1 to initialize all γ parameters, except for each residual block's last normalization layer where we initialize γ by 0 following [16] (such that the initial state of a residual block is identity). We use a weight decay of 0.0001 for all weight layers, including γ and β (following [17] but unlike [20, 16]). We train 100 epochs for all models, and decrease the learning rate by 10× at 30, 60, and 90 epochs. During training, we adopt the data augmentation of [58] as implemented by [17]. We evaluate the top-1 classification error on the center crops of 224×224 pixels in the validation set. To reduce random variations, we report the median error rate of the final 5 epochs [16]. Other implementation details follow [17].

Our baseline is the ResNet trained with BN [20]. To compare with LN, IN, and GN, we replace BN with the specific variant. We use the same hyper-parameters for all models. We set $G = 32$ for GN by default.

Comparison of feature normalization methods. We first experiment with a regular batch size of **32 images** (per GPU) [26, 20]. BN works successfully in this regime, so this is a strong baseline to compare with. Figure 4 shows the error curves, and Table 1 shows the final results.

Figure 4 shows that *all* of these normalization methods are able to converge. LN has a small degradation of 1.7%

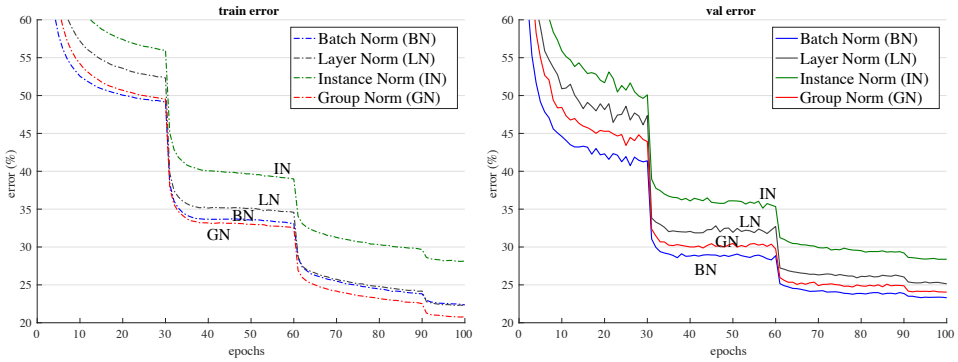


Figure 4. **Comparison of error curves** with a batch size of **32 images**/GPU. We show the ImageNet training error (left) and validation error (right) *vs*. numbers of training epochs. The model is ResNet-50.

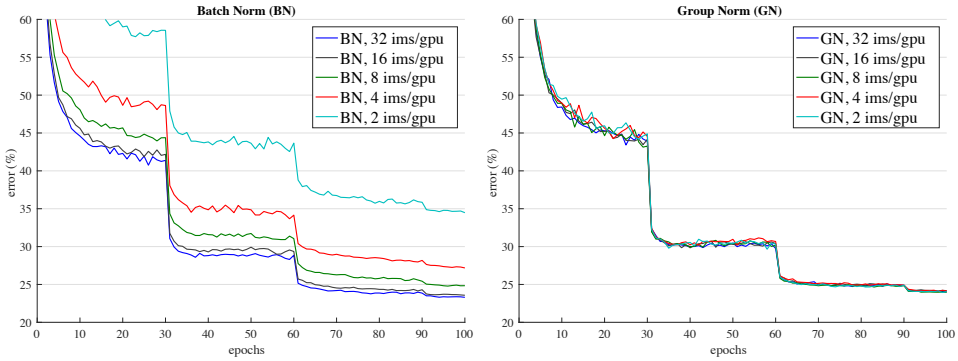


Figure 5. **Sensitivity to batch sizes**: ResNet-50's validation error of BN (left) and GN (right) trained with 32, 16, 8, 4, and 2 images/GPU.

	BN	LN	IN	GN
val error	**23.6**	25.3	28.4	24.1
△ (*vs*. BN)	-	*1.7*	*4.8*	***0.5***

Table 1. **Comparison of error rates** (%) of ResNet-50 in the ImageNet validation set, trained with a batch size of **32 images**/GPU. The error curves are in Figure 4.

comparing with BN. This is an encouraging result, as it suggests that normalizing along *all* channels (as done by LN) of a *convolutional* network is reasonably good. IN also makes the model converge, but is 4.8% worse than BN.[3]

In this regime where BN works well, GN is able to approach BN's accuracy, with a decent degradation of 0.5% in the validation set. Actually, Figure 4 (left) shows that GN has *lower training error* than BN, indicating that GN is effective for easing *optimization*. The slightly higher validation error of GN implies that GN loses some regularization ability of BN. This is understandable, because BN's mean and variance computation introduces uncertainty caused by the stochastic batch sampling, which helps regularization [26]. This uncertainty is missing in GN (and LN/IN). But it is possible that GN combined with a suitable regularizer will improve results. This can be a future research topic.

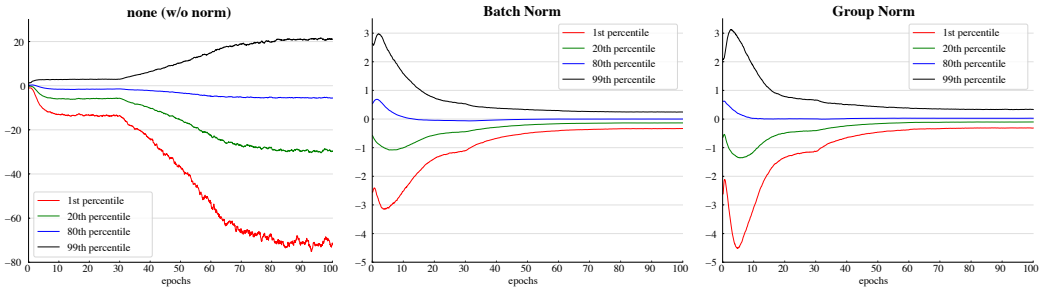
[3]For completeness, we have also trained ResNet-50 with WN [51], which is *filter* (instead of *feature*) normalization. WN's result is 28.2%.

batch size	32	16	8	4	2
BN	**23.6**	**23.7**	24.8	27.3	34.7
GN	24.1	24.2	**24.0**	**24.2**	**24.1**
△	*0.5*	*0.5*	*-0.8*	*-3.1*	*-10.6*

Table 2. **Sensitivity to batch sizes**. We show ResNet-50's validation error (%) in ImageNet. The last row shows the differences between BN and GN. The error curves are in Figure 5. This table is visualized in Figure 1.

Small batch sizes. Although BN benefits from the stochasticity under some situations, its error increases when the batch size becomes smaller and the uncertainty gets bigger. We show this in Figure 1, Figure 5, and Table 2.

We evaluate batch sizes of 32, 16, 8, 4, 2 images per GPU. In all cases, the BN mean and variance are computed within each GPU and not synchronized. All models are trained in 8 GPUs. In this set of experiments, we adopt the linear learning rate scaling rule [31, 4, 16] to adapt to batch size changes — we use a learning rate of 0.1 [20] for the batch size of 32, and $0.1N/32$ for a batch size of N. This linear scaling rule works well for BN if the total batch size changes (by changing the number of GPUs) but the per-GPU batch size does not change [16]. We keep the same number of training epochs for all cases (Figure 5, x-axis). All other hyper-parameters are unchanged.



	error
none	29.2
BN	28.0
GN	**27.6**

Figure 6. **Evolution of feature distributions** of conv_{5_3}'s output (before normalization and ReLU) from VGG-16, shown as the $\{1, 20, 80, 99\}$ percentile of responses. The table on the right shows the ImageNet validation error (%). Models are trained with 32 images/GPU.

# groups (G)						
64	32	16	8	4	2	1 (=LN)
24.6	**24.1**	24.6	24.4	24.6	24.7	25.3
0.5	-	*0.5*	*0.3*	*0.5*	*0.6*	*1.2*

# channels per group						
64	32	16	8	4	2	1 (=IN)
24.4	24.5	**24.2**	24.3	24.8	25.6	28.4
0.2	*0.3*	-	*0.1*	*0.6*	*1.4*	*4.2*

Table 3. **Group division**. We show ResNet-50's validation error (%) in ImageNet, trained with 32 images/GPU. (Top): a given number of groups. (Bottom): a given number of channels per group. The last rows show the differences with the best number.

Figure 5 (left) shows that BN's error becomes considerably higher with small batch sizes. GN's behavior is more stable and insensitive to the batch size. Actually, Figure 5 (right) shows that GN has very similar curves (subject to random variations) across a wide range of batch sizes from 32 to 2. In the case of a batch size of 2, GN has **10.6%** lower error rate than its BN counterpart (24.1% *vs*. 34.7%).

These results indicate that the batch mean and variance estimation can be overly stochastic and inaccurate, especially when they are computed over 4 or 2 images. However, this stochasticity disappears if the statistics are computed from 1 image, in which case BN becomes similar to IN at training time. We see that IN has a better result (28.4%) than BN with a batch size of 2 (34.7%).

The robust results of GN in Table 2 demonstrate GN's strength. It allows to remove the batch size constraint imposed by BN, which can give considerably more memory (*e.g.*, 16× or more). This will make it possible to train higher-capacity models that would be otherwise bottlenecked by memory limitation. We hope this will create new opportunities in architecture design.

Comparison with Batch Renorm (BR). BR [25] introduces two extra parameters (r and d in [25]) that constrain the estimated mean and variance of BN. Their values are controlled by r_{max} and d_{max}. To apply BR to ResNet-50, we have carefully chosen these hyper-parameters, and found that $r_{\text{max}} = 1.5$ and $d_{\text{max}} = 0.5$ work best for ResNet-50. With a batch size of 4, ResNet-50 trained with BR has an error rate of 26.3%. This is better than BN's 27.3%, but still 2.1% higher than GN's 24.2%.

Group division. Thus far all presented GN models are trained with a group number of $G = 32$. Next we evaluate different ways of dividing into groups. With a given fixed group number, GN performs reasonably well for all values of G we studied (Table 3, top panel). In the extreme case of $G = 1$, GN is equivalent to LN, and its error rate is higher than all cases of $G > 1$ studied.

We also evaluate fixing the number of channels per group (Table 3, bottom panel). Note that because the layers can have different channel numbers, the group number G can change across layers in this setting. In the extreme case of 1 channel per group, GN is equivalent to IN. Even if using as few as 2 channels per group, GN has substantially lower error than IN (25.6% *vs*. 28.4%). This result shows the effect of grouping channels when performing normalization.

Deeper models. We have also compared GN with BN on ResNet-101 [20]. With a batch size of 32, our BN baseline of ResNet-101 has 22.0% validation error, and the GN counterpart has 22.4%, slightly worse by 0.4%. With a batch size of 2, GN ResNet-101's error is 23.0%. This is still a decently stable result considering the very small batch size, and it is 8.9% better than the BN counterpart's 31.9%.

Results and analysis of VGG models. To study GN/BN compared to *no normalization*, we consider VGG-16 [56] that can be healthily trained without normalization layers. We apply BN or GN right after each convolutional layer. Figure 6 shows the evolution of the feature distributions of conv_{5_3} (the last convolutional layer). GN and BN behave *qualitatively similar*, while being substantially different with the variant that uses no normalization; this phenomenon is also observed for all other convolutional layers. This comparison suggests that performing normalization is essential for controlling the distribution of features.

For VGG-16, GN is *better* than BN by 0.4% (Figure 6, right). This possibly implies that VGG-16 benefits less from BN's regularization effect, and GN (that leads to lower training error) is superior to BN in this case.

4.2. Object Detection and Segmentation in COCO

Next we evaluate fine-tuning the models for transferring to object detection and segmentation. These computer vision tasks in general benefit from higher-resolution input, so the batch size tends to be small in common practice (1 or 2 images/GPU [12, 47, 18, 36]). As a result, BN is turned into a *linear* layer $y = \frac{\gamma}{\sigma}(x - \mu) + \beta$ where μ and σ are pre-computed from the pre-trained model and frozen [20]. We denote this as BN*, which in fact performs no normalization during fine-tuning. We have also tried a variant that fine-tunes BN (normalization is performed and not frozen) and found it works poorly (reducing ~6 AP with a batch size of 2), so we ignore this variant.

We experiment on the Mask R-CNN baselines [18], implemented in the publicly available codebase of *Detectron* [13]. We use the end-to-end variant with the same hyper-parameters as in [13]. We replace BN* with GN during fine-tuning, using the corresponding models pre-trained from ImageNet.[4] During fine-tuning, we use a weight decay of 0 for the γ and β parameters, which is important for good detection results when γ and β are being tuned. We fine-tune with a batch size of 1 image/GPU and 8 GPUs.

The models are trained in the COCO `train2017` set and evaluated in the COCO `val2017` set (a.k.a `minival`). We report the standard COCO metrics of Average Precision (AP), AP_{50}, and AP_{75}, for bounding box detection (AP^{bbox}) and instance segmentation (AP^{mask}).

Results of C4 backbone. Table 4 shows the comparison of GN *vs*. BN* on Mask R-CNN using a $conv_4$ backbone ("C4" [18]). This C4 variant uses ResNet's layers of up to $conv_4$ to extract feature maps, and ResNet's $conv_5$ layers as the Region-of-Interest (RoI) heads for classification and regression. As they are inherited from the pre-trained model, the backbone and head both involve normalization layers.

On this baseline, GN improves over BN* by 1.1 box AP and 0.8 mask AP. We note that the pre-trained GN model is slightly worse than BN in ImageNet (24.1% *vs*. 23.6%), but GN still outperforms BN* for fine-tuning. BN* creates inconsistency between pre-training and fine-tuning (frozen), which may explain the degradation.

We have also experimented with the LN variant, and found it is 1.9 box AP worse than GN and 0.8 worse than BN*. Although LN is also independent of batch sizes, its representational power is weaker than GN.

Results of FPN backbone. Next we compare GN and BN* on Mask R-CNN using a Feature Pyramid Network (FPN) backbone [35], the currently state-of-the-art framework in COCO. Unlike the C4 variant, FPN exploits all pre-trained layers to construct a pyramid, and appends randomly initialized layers as the head. In [35], the box head consists of two hidden fully-connected layers (2fc). We find that replacing the 2fc box head with 4conv1fc (similar to [48]) can better leverage GN. The resulting comparisons are in Table 5.

As a baseline, BN* has 38.6 box AP using the 4conv1fc head, on par with its 2fc counterpart using the same pre-trained model (38.5 AP). By adding GN to all convolutional layers of the box head (but still using the BN* backbone), we increase the box AP by 0.9 to 39.5 (2nd row, Table 5). This ablation shows that a substantial portion of GN's improvement for detection is from *normalization in the head* (which is also done by the C4 variant). On the contrary, applying BN to the box head (that has 512 RoIs per image) does not provide satisfactory result and is ~9 AP worse — in detection, the batch of RoIs are sampled from the same image and their distribution is not *i.i.d.*, and the *non-i.i.d.* distribution is also an issue that degrades BN's batch statistics estimation [25]. GN does not suffer from this problem.

Next we replace the FPN backbone with the GN-based counterpart, *i.e.*, the GN pre-trained model is used during fine-tuning (3rd row, Table 5). Applying GN to the backbone *alone* contributes a 0.5 AP gain (from 39.5 to 40.0), suggesting that GN helps when transferring features.

backbone	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
BN*	37.7	57.9	40.9	32.8	54.3	34.7
GN	**38.8**	**59.2**	**42.2**	**33.6**	**55.9**	**35.4**

Table 4. Detection and segmentation **ablation results in COCO**, using Mask R-CNN with **ResNet-50 C4**. BN* means BN is frozen.

backbone	box head	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
BN*	-	38.6	59.5	41.9	34.2	56.2	36.1
BN*	GN	39.5	60.0	43.2	34.4	56.4	**36.3**
GN	GN	**40.0**	**61.0**	**43.3**	**34.8**	**57.3**	**36.3**

Table 5. Detection and segmentation **ablation results in COCO**, using Mask R-CNN with **ResNet-50 FPN** and a 4conv1fc bounding box head. BN* means BN is frozen.

	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
R50 BN*	38.6	59.8	42.1	34.5	56.4	36.3
R50 GN	40.3	61.0	44.0	35.7	57.9	37.7
R50 GN, long	**40.8**	**61.6**	**44.4**	**36.1**	**58.5**	**38.2**
R101 BN*	40.9	61.9	44.8	36.4	58.5	38.7
R101 GN	41.8	62.5	45.4	36.8	59.2	39.0
R101 GN, long	**42.3**	**62.8**	**46.2**	**37.2**	**59.7**	**39.5**

Table 6. **Detection and segmentation results in COCO** using Mask R-CNN and FPN. Here BN* is the default Detectron baseline [13], and GN is applied to the backbone, box head, and mask head. "long" means training with more iterations. Code of these results are in https://github.com/facebookresearch/Detectron/blob/master/projects/GN.

[4] Detectron [13] uses pre-trained models provided by the authors of [20]. For fair comparisons, we instead use the models pre-trained in this paper. The object detection and segmentation accuracy is statistically similar between these pre-trained models.

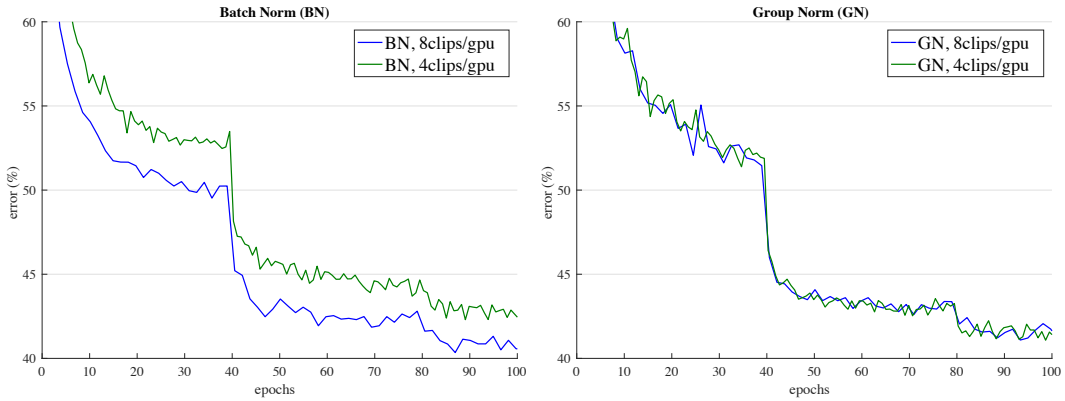


Figure 7. **Error curves in Kinetics with an input length of 32 frames**. We show ResNet-50 I3D's validation error of BN (left) and GN (right) using a batch size of 8 and 4 clips/GPU. The monitored validation error is the 1-clip error under the same data augmentation as the training set, while the final validation accuracy in Table 8 is 10-clip testing without data augmentation.

from scratch	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
R50 BN [34]	34.5	55.2	37.7	-	-	-
R50 GN	39.5	59.8	43.6	35.2	56.9	37.6
R101 GN	41.0	61.1	44.9	36.4	58.2	38.7

Table 7. Detection and segmentation results trained **from scratch** in COCO using Mask R-CNN and FPN. Here the BN results are from [34], and BN is synced across GPUs [43] and is *not* frozen. Code of these results are in https://github.com/facebookresearch/Detectron/blob/master/projects/GN.

Table 6 shows the full results of GN (applied to the backbone, box head, and mask head), compared with the standard Detectron baseline [13] based on BN^*. Using the same hyper-parameters as [13], GN increases over BN^* by a healthy margin. Moreover, we found that GN is not fully trained with the default schedule in [13], so we also tried increasing the iterations from 180k to 270k (BN^* does not benefit from longer training). Our final ResNet-50 GN model ("long", Table 6) is **2.2** points box AP and **1.6** points mask AP better than its BN^* variant.

Training Mask R-CNN from scratch. GN allows us to easily investigate training object detectors *from scratch* (without any pre-training). We show the results in Table 7, where the GN models are trained for 270k iterations.[5] To our knowledge, our numbers (**41.0** box AP and **36.4** mask AP) are the best *from-scratch* results in COCO reported to date; they can even compete with the ImageNet-pretrained results in Table 6. As a reference, with synchronous BN [43], a concurrent work [34] achieves a from-scratch result of 34.5 box AP using R50 (Table 7), and 36.3 using a specialized backbone.

[5]For models trained from scratch, we turn off the default StopGrad in Detectron that freezes the first few layers.

clip length	32	32	64
batch size	8	4	4
BN	**73.3 / 90.7**	72.1 / 90.0	73.3 / 90.8
GN	73.0 / 90.6	**72.8 / 90.6**	**74.5 / 91.7**

Table 8. **Video classification results in Kinetics**: ResNet-50 I3D baseline's top-1 / top-5 accuracy (%).

4.3. Video Classification in Kinetics

Lastly we evaluate video classification in the Kinetics dataset [30]. Many video classification models [60, 6] extend the features to 3D spatial-temporal dimensions. This is memory-demanding and imposes constraints on the batch sizes and model designs.

We experiment with Inflated 3D (I3D) convolutional networks [6]. We use the ResNet-50 I3D *baseline* as described in [62]. The models are pre-trained from ImageNet. For both BN and GN, we extend the normalization from over (H, W) to over (T, H, W), where T is the temporal axis. We train in the 400-class Kinetics training set and evaluate in the validation set. We report the top-1 and top-5 classification accuracy, using standard 10-clip testing that averages softmax scores from 10 clips regularly sampled.

We study two different temporal lengths: 32-frame and 64-frame input clips. The 32-frame clip is regularly sampled with a frame interval of 2 from the raw video, and the 64-frame clip is sampled continuously. The model is fully convolutional in spacetime, so the 64-frame variant consumes about $2\times$ more memory. We study a batch size of 8 or 4 clips/GPU for the 32-frame variant, and 4 clips/GPU for the 64-frame variant due to memory limitation.

Results of 32-frame inputs. Table 8 (col. 1, 2) shows the video classification accuracy in Kinetics using 32-frame clips. For the batch size of 8, GN is slightly worse than BN by 0.3% top-1 accuracy and 0.1% top-5. This shows that GN is competitive with BN when BN works well. For

the smaller batch size of 4, GN's accuracy is kept similar (72.8 / 90.6 *vs*. 73.0 / 90.6), but is better than BN's 72.1 / 90.0. BN's accuracy is decreased by 1.2% when the batch size decreases from 8 to 4.

Figure 7 shows the error curves. BN's error curves (left) have a noticeable gap when the batch size decreases from 8 to 4, while GN's error curves (right) are very similar.

Results of 64-frame inputs. Table 8 (col. 3) shows the results of using 64-frame clips. In this case, BN has a result of 73.3 / 90.8. These appear to be acceptable numbers (*vs*. 73.3 / 90.7 of 32-frame, batch size 8), but *the trade-off between the temporal length (64* vs. *32) and batch size (4* vs. *8) could have been overlooked*. Comparing col. 3 and col. 2 in Table 8, we find that the temporal length actually has positive impact (+1.2%), but it is veiled by BN's negative effect of the smaller batch size.

GN does not suffer from this trade-off. The 64-frame variant of GN has 74.5 / 91.7 accuracy, showing healthy gains over its BN counterpart and all BN variants. GN helps the model benefit from temporal length, and the longer clip boosts the top-1 accuracy by 1.7% (top-5 1.1%) with the same batch size.

The improvement of GN on detection, segmentation, and video classification demonstrates that GN is a strong alternative to the powerful and currently dominant BN technique in these tasks.

5. Discussion and Future Work

We have presented GN as an effective normalization layer without exploiting the batch dimension. We have evaluated GN's behaviors in a variety of applications. We note, however, that BN has been so influential that many state-of-the-art systems and their hyper-parameters have been designed for it, which may not be optimal for GN-based models. It is possible that re-designing the systems or searching new hyper-parameters for GN will give better results.

In addition, we have shown that GN is related to LN and IN, two normalization methods that are particularly successful in training recurrent (RNN/LSTM) or generative (GAN) models. This suggests us to study GN in those areas in the future. We will also investigate GN's performance on learning representations for reinforcement learning (RL) tasks, *e.g*., [54], where BN is playing an important role for training very deep models [20].

Acknowledgement. We would like to thank Piotr Dollár and Ross Girshick for helpful discussions.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *Operating Systems Design and Implementation (OSDI)*, 2016.

[2] D. Arpit, Y. Zhou, B. Kota, and V. Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. In *ICML*, 2016.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

[4] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *arXiv:1606.04838*, 2016.

[5] M. Carandini and D. J. Heeger. Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 2012.

[6] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[7] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[8] T. Cohen and M. Welling. Group equivariant convolutional networks. In *ICML*, 2016.

[9] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[10] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *NIPS*, 2012.

[11] S. Dieleman, J. De Fauw, and K. Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *ICML*, 2016.

[12] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[13] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. https://github.com/facebookresearch/detectron, 2018.

[14] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[16] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv:1706.02677*, 2017.

[17] S. Gross and M. Wilber. Training and investigating Residual Nets. https://github.com/facebook/fb.resnet.torch, 2016.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[21] D. J. Heeger. Normalization of cell responses in cat striate cortex. *Visual neuroscience*, 1992.

[22] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[23] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.

[24] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[25] S. Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *NIPS*, 2017.

[26] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[27] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[28] K. Jarrett, K. Kavukcuoglu, Y. LeCun, et al. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.

[29] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[30] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The Kinetics human action video dataset. *arXiv:1705.06950*, 2017.

[31] A. Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv:1404.5997*, 2014.

[32] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[33] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*. 1998.

[34] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun. DetNet: A backbone network for object detection. *arXiv:1804.06215*, 2018.

[35] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[36] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*. 2014.

[38] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[39] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[40] S. Lyu and E. P. Simoncelli. Nonlinear image representation using divisive normalization. In *CVPR*, 2008.

[41] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001.

[42] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[43] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun. MegDet: A large mini-batch object detector. In *CVPR*, 2018.

[44] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[45] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *NIPS*, 2017.

[46] M. Ren, R. Liao, R. Urtasun, F. H. Sinz, and R. S. Zemel. Normalizing the normalizers: Comparing and extending network normalization schemes. In *ICLR*, 2017.

[47] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[48] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *TPAMI*, 2017.

[49] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 1986.

[50] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[51] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016.

[52] O. Schwartz and E. P. Simoncelli. Natural signal statistics and sensory gain control. *Nature neuroscience*, 2001.

[53] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

[54] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 2017.

[55] E. P. Simoncelli and B. A. Olshausen. Natural image statistics and neural representation. *Annual review of neuroscience*, 2001.

[56] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[57] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. In *ICLR Workshop*, 2016.

[58] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[59] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[60] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015.

[61] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016.

[62] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018.

[63] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[64] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In *ECCV*, 2014.

[65] X. Zhang, X. Zhou, M. Lin, and J. Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

A Closed-form Solution to Photorealistic Image Stylization

Yijun Li[1], Ming-Yu Liu[2], Xueting Li[1], Ming-Hsuan Yang[1,2], Jan Kautz[2]

[1]University of California, Merced [2]NVIDIA
{yli62,xli75,mhyang}@ucmerced.edu {mingyul,jkautz}@nvidia.com

Abstract. Photorealistic image stylization concerns transferring style of a reference photo to a content photo with the constraint that the stylized photo should remain photorealistic. While several photorealistic image stylization methods exist, they tend to generate spatially inconsistent stylizations with noticeable artifacts. In this paper, we propose a method to address these issues. The proposed method consists of a stylization step and a smoothing step. While the stylization step transfers the style of the reference photo to the content photo, the smoothing step ensures spatially consistent stylizations. Each of the steps has a closed-form solution and can be computed efficiently. We conduct extensive experimental validations. The results show that the proposed method generates photorealistic stylization outputs that are more preferred by human subjects as compared to those by the competing methods while running much faster. Source code and additional results are available at https://github.com/NVIDIA/FastPhotoStyle.

Keywords: Image stylization, photorealism, closed-form solution.

1 Introduction

Photorealistic image stylization aims at changing style of a photo to that of a reference photo. For a faithful stylization, content of the photo should remain the same. Furthermore, the output photo should look like a real photo as it were captured by a camera. Figure 1 shows two photorealistic image stylization examples. In one example, we transfer a summery photo to a snowy one, while in the other, we transfer a day-time photo to a night-time photo.

Classical photorealistic stylization methods are mostly based on color/tone matching [1,2,3,4] and are often limited to specific scenarios (e.g., seasons [5] and headshot portraits [6]). Recently, Gatys et al. [7,8] show that the correlations between deep features encode the visual style of an image and propose an optimization-based method, the neural style transfer algorithm, for image stylization. While the method shows impressive performance for *artistic* stylization (converting images to paintings), it often introduces structural artifacts and distortions when applied to photorealistic image stylization as shown in Figure 1(c). In a follow-up work, Luan et al. [9] propose adding a regularization term to the optimization objective function of the neural style transfer algorithm

(a) Style (b) Content (c) Gatys et al. [8] (d) Luan et al. [9] (e) Ours

Fig. 1: Given a style photo (a) and a content photo (b), photorealistic image stylization aims at transferring style of the style photo to the content photo as shown in (c), (d) and (e). Comparing with existing methods [8,9], the output photos computed by our method are stylized more consistently and with fewer artifacts. Moreover, our method runs an order of magnitude faster.

for avoiding distortions in the stylization output. However, this often results in inconsistent stylizations in semantically uniform regions as shown in Figure 1(d). To address the issues, we propose a photorealistic image stylization method.

Our method consists of a stylization step and a smoothing step. Both have a closed-form solution[1] and can be computed efficiently. The stylization step is based on the whitening and coloring transform (WCT) [10], which stylizes images via feature projections. The WCT was designed for *artistic* stylization. Similar to the neural style transfer algorithm, it suffers from structural artifacts when applied to photorealistic image stylization. Our WCT-based stylization step resolves the issue by utilizing a novel network design for feature transform. The WCT-based stylization step alone may generate spatially inconsistent stylizations. We resolve this issue by the proposed smoothing step, which is based on a manifold ranking algorithm. We conduct extensive experimental validation with comparison to the state-of-the-art methods. User study results show that our method generates outputs with better stylization effects and fewer artifacts.

2 Related Work

Existing stylization methods can be classified into two categories: global and local. Global methods [1,2,11] achieve stylization through matching the means and variances of pixel colors [1] or their histograms [2]. Local methods [12,6,13,5,14] stylize images through finding dense correspondences between the content and style photos based on either low-level or high-level features. These approaches are slow in practice. Also, they are often developed for specific scenarios (e.g., day-time or season change).

[1] A closed-form solution means that the solution can be obtained in a fixed finite number of operations, including convolutions, max-pooling, whitening, etc.

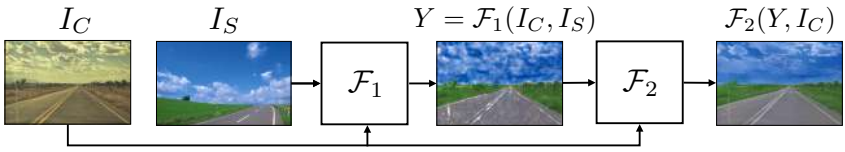


Fig. 2: Our photorealistic image stylization method consists of two closed-form steps: $\mathcal{F}_1$ and $\mathcal{F}_2$. While $\mathcal{F}_1$ maps I_C to an intermediate image carrying the style of I_S, $\mathcal{F}_2$ removes noticeable artifacts, which produces a photorealistic output.

Gatys et al. [7,8] propose the neural style transfer algorithm for *artistic* stylization. The major step in the algorithm is to solve an optimization problem of matching the Gram matrices of deep features extracted from the content and style photos. A number of methods have been developed [15,16,17,18,19,20,21,22,10,23] to further improve its stylization performance and speed. However, these methods do not aim for preserving photorealism (see Figure 1(c)). Post-processing techniques [24,25] have been proposed to refine these results by matching the gradients between the input and output photos.

Photorealistic image stylization is related to the image-to-image translation problem [26,27,28,29,30,31,32,33] where the goal is to learn to translate an image from one domain to another. However, photorealistic image stylization does not require a training dataset of content and style images for learning the translation function. Photorealistic image stylization can be considered as a special kind of image-to-image translation. Not only can it be used to translate a photo to a different domain (e.g., form day to night-time) but also transfer style (e.g., extent of darkness) of a specific reference image to the content image.

Closest to our work is the method of Luan et al. [9]. It improves photorealism of stylization outputs computed by the neural style transfer algorithm [7,8] by incorporating a new loss term to the optimization objective, which has the effect of better preserving local structures in the content photo. However, it often generates inconsistent stylization with noticeable artifacts (Figure 1(d)). Moreover, the method is computationally expensive. Our proposed algorithm aims at efficient and effective photorealistic image stylization. We demonstrate that it performs favorably against Luan et al. [9] in terms of both quality and speed.

3 Photorealistic Image Stylization

Our photorealistic image stylization algorithm consists of two steps as illustrated in Figure 2. The first step is a stylization transform $\mathcal{F}_1$ called PhotoWCT. Given a style photo I_S, $\mathcal{F}_1$ transfer the style of I_S to the content photo I_C while minimizing structural artifacts in the output image. Although $\mathcal{F}_1$ can faithfully stylize I_C, it often generates inconsistent stylizations in semantically similar regions. Therefore, we use a photorealistic smoothing function $\mathcal{F}_2$, to eliminate these

artifacts. Our whole algorithm can be written as a two-step mapping function:

$$\mathcal{F}_2\Big(\mathcal{F}_1(I_C, I_S), I_C\Big), \tag{1}$$

In the following, we discuss the stylization and smoothing steps in details.

3.1 Stylization

The PhotoWCT is based on the WCT [10]. It utilizes a novel network design for achieving photorealistic image stylization. We briefly review the WCT below.

WCT. The WCT [10] formulates stylization as an image reconstruction problem with feature projections. To utilize WCT, an auto-encoder for general image reconstruction is first trained. Specifically, it uses the VGG-19 model [34] as the encoder $\mathcal{E}$ (weights are kept fixed) and trains a decoder $\mathcal{D}$ for reconstructing the input image. The decoder is symmetrical to the encoder and uses upsampling layers (pink blocks in Figure 3(a)) to enlarge the spatial resolutions of the feature maps. Once the auto-encoder is trained, a pair of projection functions are inserted at the network bottleneck to perform stylization through the whitening (P_C) and coloring (P_S) transforms. The key idea behind the WCT is to directly match feature correlations of the content image to those of the style image via the two projections. Specifically, given a pair of content image I_C and style image I_S, the WCT first extracts their vectorised VGG features $H_C = \mathcal{E}(I_C)$ and $H_S = \mathcal{E}(I_S)$, and then transform the content feature H_C via

$$H_{CS} = P_S P_C H_C, \tag{2}$$

where $P_C = E_C \Lambda_C^{-\frac{1}{2}} E_C^\top$, and $P_S = E_S \Lambda_S^{\frac{1}{2}} E_S^\top$. Here Λ_C and Λ_S are the diagonal matrices with the eigenvalues of the covariance matrix $H_C H_C^\top$ and $H_S H_S^\top$ respectively. The matrices E_C and E_S are the corresponding orthonormal matrices of the eigenvectors, respectively. After the transformation, the correlations of transformed features match those of the style features, i.e., $H_{CS} H_{CS}^\top = H_S H_S^\top$. Finally, the stylized image is obtained by directly feeding the transformed feature map into the decoder: $Y = \mathcal{D}(H_{CS})$. For better stylization performance, Li et al. [10] use a multi-level stylization strategy, which performs the WCT on the VGG features at different layers.

The WCT performs well for artistic image stylization. However it generates structural artifacts (e.g., distortions on object boundaries) for photorealistic image stylization (Figure 4(c)). The proposed PhotoWCT is designed to suppress these structural artifacts.

PhotoWCT. Our PhotoWCT design is motivated by the observation that the max-pooling operation in the WCT reduces spatial information in feature maps. Simply upsampling feature maps in the decoder fails to recover detailed structures of the input image. That is, we need to pass the lost spatial information to the decoder to facilitate reconstructing these fine details. Inspired by the success of

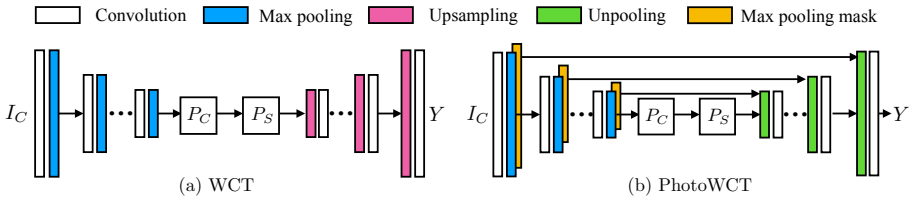


Fig. 3: The PhotoWCT and WCT share the same encoder architecture and projection steps. In the PhotoWCT, we replace the upsampling layers (pink) with unpooling layers (green). Note that the unpooling layer is used together with the pooling mask (yellow) which records *where* carries the *maximum* over each max pooling region in the corresponding pooling layer [35].

the unpooling layer [35,36,37] in preserving spatial information, the PhotoWCT replaces the upsampling layers in the WCT with unpooling layers. The PhotoWCT function is formulated as

$$Y = \mathcal{F}_1(I_C, I_S) = \overline{\mathcal{D}}(P_S P_C H_C), \tag{3}$$

where $\overline{\mathcal{D}}$ is the decoder, which contains unpooling layers and is trained for image reconstruction. Figure 3 illustrates the network architecture difference between the WCT and the proposed PhotoWCT.

Figure 4(c) and (d) compare the stylization results of the WCT and PhotoWCT. As highlighted in close-ups, the straight lines along the building boundary in the content image becomes zigzagged in the WCT stylization result but remains straight in the PhotoWCT result. The PhotoWCT-stylized image has much fewer structural artifacts. We also perform a user study in the experiment section to quantitatively verify that the PhotoWCT generally leads to better stylization effects than the WCT.

3.2 Photorealistic Smoothing

The PhotoWCT-stylized result (Figure 4(d)) still looks less like a photo since semantically similar regions are often stylized inconsistently. As shown in Figure 4, when applying the PhotoWCT to stylize the day-time photo using the night-time photo, the stylized sky region would be more photorealistic if it were uniformly dark blue instead of partly dark and partly light blue. It is based on this observation, we employ the pixel affinities in the content photo to smooth the PhotoWCT-stylized result.

We aim to achieve two goals in the smoothing step. First, pixels with similar content in a local neighborhood should be stylized similarly. Second, the output should not deviate significantly from the PhotoWCT result in order to maintain the global stylization effects. We first represent all pixels as nodes in a graph and define an affinity matrix $W = \{w_{ij}\} \in \mathbb{R}^{N \times N}$ (N is the number of pixels) to

(a) Style (b) Content

(c) WCT [10] (d) PhotoWCT

(e) WCT + smoothing (f) PhotoWCT + smoothing

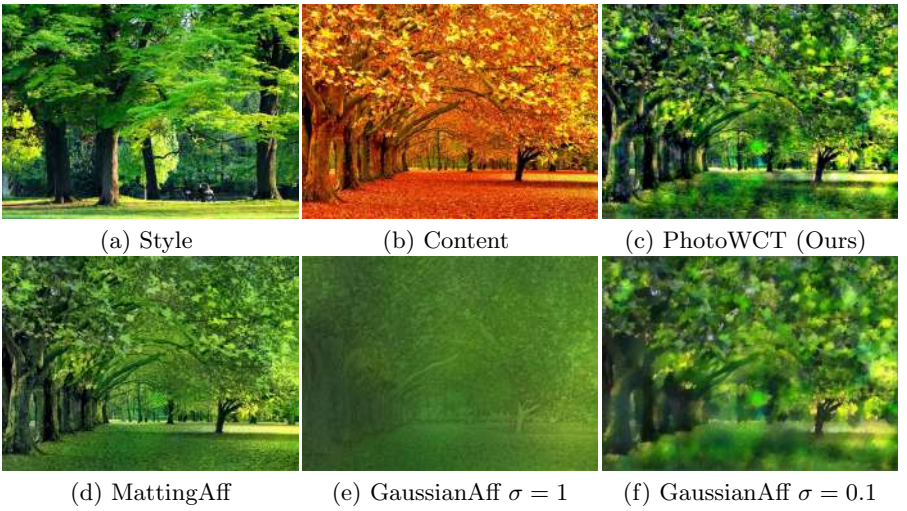
Fig. 4: The stylization output generated by the PhotoWCT better preserves local structures in the content images, which is important for the image smoothing step as shown in (e) and (f).

describe pixel similarities. We define a smoothness term and a fitting term that model these two goals in the following optimization problem:

$$\underset{r}{\operatorname{argmin}} \frac{1}{2}\Big(\sum_{i,j=1}^{N} w_{ij}\Big\|\frac{r_i}{\sqrt{d_{ii}}} - \frac{r_j}{\sqrt{d_{jj}}}\Big\|^2 + \lambda \sum_{i=1}^{N}\|r_i - y_i\|^2\Big), \tag{4}$$

where y_i is the pixel color in the PhotoWCT-stylized result Y and r_i is the pixel color in the desired smoothed output R. The variable $d_{ii} = \sum_j w_{ij}$ is the diagonal element in the degree matrix D of W, i.e., $D = \text{diag}\{d_{11}, d_{22}, ..., d_{NN}\}$. In (4), λ controls the balance of the two terms.

Our formulation is motivated by the graph-based ranking algorithms [38,39]. In the ranking algorithms, Y is a binary input where each element indicates if a specific item is a query ($y_i = 1$ if y_i is a query and $y_i = 0$ otherwise). The optimal solution R is the ranking values of all the items based on their pairwise affinities. In our method, we set Y as the PhotoWCT-stylized result. The optimal solution R is the smoothed version of Y based on the pairwise pixel affinities,



(a) Style (b) Content (c) PhotoWCT (Ours)

(d) MattingAff (e) GaussianAff $\sigma = 1$ (f) GaussianAff $\sigma = 0.1$

Fig. 5: Smoothing with different affinities. To refine the PhotoWCT result in (c), it is hard to find an optimal σ for the Gaussian Affinity that performs globally well as shown in (e)-(f). In contrast, using the Matting Affinity can simultaneously smooth different regions well as shown in (d).

which encourages consistent stylization within semantically similar regions. The above optimization problem is a simple quadratic problem with a closed-form solution, which is given by

$$R^* = (1-\alpha)(I-\alpha S)^{-1}Y, \tag{5}$$

where I is the identity matrix, $\alpha = \frac{1}{1+\lambda}$ and S is the normalized Laplacian matrix computed from I_C, i.e., $S = D^{-\frac{1}{2}}WD^{-\frac{1}{2}} \in \mathbb{R}^{N\times N}$. As the constructed graph is often sparsely connected (i.e., most elements in W are zero), the inverse operation in (5) can be computed efficiently. With the closed-form solution, the smoothing step can be written as a function mapping given by:

$$R^* = \mathcal{F}_2(Y, I_C) = (1-\alpha)(I-\alpha S)^{-1}Y. \tag{6}$$

Affinity. The affinity matrix W is computed using the content photo based on an 8-connected image graph assumption. While several choices of affinity metrics exist, a popular one is to define the affinity (denoted as GaussianAff) as $w_{ij} = e^{-\|I_i - I_j\|^2/\sigma^2}$ where I_i, I_j are the RGB values of adjacent pixels i, j and σ is a global scaling hyper-parameter [40]. However, it is difficult to determine the σ value in practice. It often results in either over-smoothing the entire photo (Figure 5(e)) or stylizing the photo inconsistently (Figure 5(f)). To avoid selecting one global scaling hyper-parameter, we resort to the matting affinity [41,42] (denoted as MattingAff) where the affinity between two pixels is based on means and variances of pixels in a local window. Figure 5(d) shows that the matting affinity is able to simultaneously smooth different regions well.

WCT plus Smoothing. We note that the smoothing step can also remove structural artifacts in the WCT as shown in Figure 4(e). However, it leads to unsatisfactory stylization. The main reason is that the content photo and the WCT result are severely misaligned due to spatial distortions. For example, a stylized pixel of the building in the WCT result may correspond to a pixel of the sky in the content photo. Consequently this causes wrong queries in Y for the smoothing step. This shows why we need to use the PhotoWCT to remove distortions first. Figure 4(f) shows that the combination of PhotoWCT and smoothing leads to better photorealism while still maintaining faithful stylization.

4 Experiments

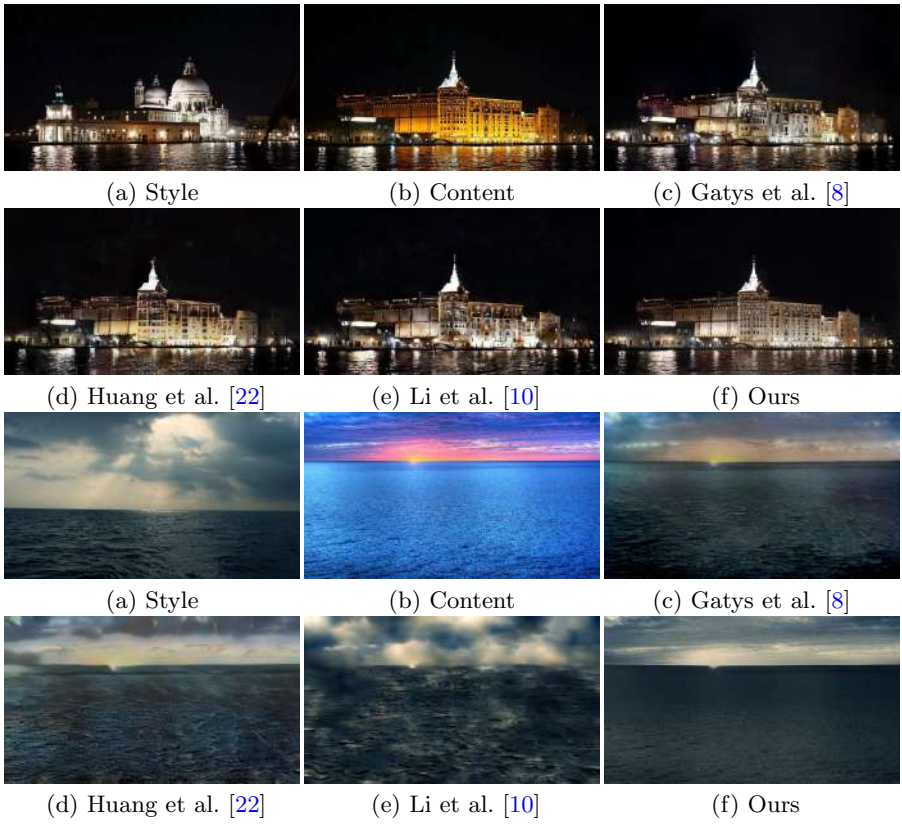
In the section, we will first discuss the implementation details. We will then present visual and user study evaluation results. Finally, we will analyze various design choices and run-time of the proposed algorithm.

Implementation details. We use the layers from *conv*1_1 to *conv*4_1 in the VGG-19 network [34] for the encoder $\mathcal{E}$. The encoder weights are given by ImageNet-pretrained weights. The decoder $\overline{\mathcal{D}}$ is the inverse of the encoder. We train the decoder by minimizing the sum of the L_2 reconstruction loss and perceptual loss [17] using the Microsoft COCO dataset [43]. We adopt the multi-level stylization strategy proposed in the WCT [10] where we apply the PhotoWCT to VGG features in different layers.

Similar to the state-of-the-art methods [44,9], our algorithm can leverage semantic label maps for obtaining better stylization results when they are available. When performing PhotoWCT stylization, for each semantic label, we compute a pair of projection matrices P_C and P_S using the features from the image regions with the same label in the content and style photos, respectively. The pair is then used to stylize these image regions. With a semantic label map, content and style matching can be performed more accurately. We note that the proposed algorithm does not need precise semantic label maps for obtaining good stylization results. Finally, we also use the efficient filtering step described in Luan et al. [9] for post-processing.

Visual comparison. We compare the proposed algorithm to two categories of stylization algorithms: photorealistic and artistic. The evaluated photorealistic stylization algorithms include Reinhard et al. [1], Pitié et al. [2], and Luan et al. [9]. Both Reinhard et al. [1] and Pitié et al. [2] represent classical techniques that are based on color statistics matching, while Luan et al. [9] is based on neural style transfer [8]. On the other hand, the set of evaluated artistic stylization algorithms include Gatys et al.[8], Huang et al.[22], and the WCT [10]. They all utilize deep networks.

Fig. 6: Visual comparisons with photorealistic stylization methods. In addition to color transfer, our method also synthesizes patterns in the style photos (e.g., the dark cloud in the top example, the snow at the bottom example).



(a) Style (b) Content (c) Gatys et al. [8]

(d) Huang et al. [22] (e) Li et al. [10] (f) Ours

(a) Style (b) Content (c) Gatys et al. [8]

(d) Huang et al. [22] (e) Li et al. [10] (f) Ours

Fig. 7: Visual comparison with artistic stylization algorithms. Note the structural distortions on object boundaries (e.g., building) and detailed edges (e.g., sea, cloud) generated by the competing stylization methods.

Figure 6 shows visual results of the evaluated photorealistic stylization algorithms. Overall, the images generated by the proposed algorithm exhibit better stylization effects. While both Reinhard et al. [1] and Pitié et al. [2] change colors of the content photos, they fail to transfer the style. We argue that photorealistic stylization cannot be purely achieved via color transfer. It requires adding new patterns that represent the style photo to the content photo. For example, in the third example of Figure 6 (bottom), our algorithm not only changes the color of ground regions to white but also synthesizes the snow patterns as they appear in the style photo. The method of Luan et al. [9] achieves good stylization effects at first glance. However, a closer look reveals that the generated photos contain noticeable artifacts, e.g., the irregular brightness on buildings and trees. Several semantically similar regions are stylized inconsistently.

Figure 7 shows the visual comparison between the proposed algorithm and artistic stylization algorithms. Although the other evaluated algorithms are able to transfer the style well, they render noticeable structural artifacts and

inconsistent stylizations across the images. In contrast, our method produces more photorealistic results.

User studies. We resort to user studies for performance evaluation since photorealistic image stylization is a highly subjective task. Our benchmark dataset consists of a set of 25 content–style pairs provided by Luan et al. [9][2]. We use the Amazon Mechanical Turk (AMT) platform for evaluation. In each question, we show the AMT workers a content–style pair and the stylized results from the evaluated algorithms displayed in random order. The AMT workers[3] are asked to select a stylized result based on the instructions. Each question is answered by 10 different workers. Hence, the performance score for each study is computed based on 250 questions. We compute the average number of times the images from an algorithm is selected, which is used as the preference score of the algorithm.

We conduct two user studies. In one study, we ask the AMT workers to select which stylized photo better carries the target style. In the other study, we ask the workers to select which stylized photo looks more like a real photo (containing fewer artifacts). Through the studies, we would like to answer which algorithm better stylizes content images and which renders better photorealistic outputs.

In Table 1, we compare the proposed algorithm to Luan et al. [9], which is the current state-of-the-art. The results show that 63.1% of the users prefer the stylization results generated by our algorithm and 73.5% regard our output photos as more photorealistic. We also compare our algorithm to the classical algorithm of Pitié et al. [2]. From Table 1, our results are as photorealistic as those computed by the classical algorithm (which simply performs color matching), and 55.2% of the users consider our stylization results better.

Table 2 compares our algorithm with the artistic stylization algorithms for user preference scores. We find our algorithm achieves a score of 56.4% and 65.6% for the stylization effect and photorealism, which are significantly better than the other algorithms. The artistic stylization algorithms do not perform well since they are not designed for the photorealistic stylization task.

WCT versus PhotoWCT. We compare the proposed algorithm with a variant where the PhotoWCT step is replaced by the WCT [10]. Again, we conduct two user studies on stylization effects and photorealism as described earlier. The result shows that the proposed algorithm is favored over its variant for better stylization 83.6% of the times and favored for better photorealism 83.2% of the times.

Sensitivity analysis on λ. In the photorealistic smoothing step, the λ balances between the smoothness term and fitting term in (4). A smaller λ renders smoother results, while a larger λ renders results that are more faithful to the queries (the PhotoWCT result). Figure 8 shows results of using different λ values. In general,

[2] We note that the user studies reported in Luan et al. [9] are based on 8 different images in their dataset, which is about one third of our benchmark dataset size.

[3] An AMT worker must have a lifetime Human Intelligent Task (HIT) approval rate greater than 98% to qualify answering the questions.

Table 1: User preference: proposed vs. Luan et al. and proposed vs. Pitié et al.

	Luan et al. [9] / proposed	Pitié et al. [2] / proposed
Better stylization	36.9% / **63.1%**	44.8% / **55.2%**
Fewer artifacts	26.5% / **73.5%**	48.8% / **51.2%**

Table 2: User preference: proposed versus *artistic* stylization algorithms.

	Gatys et al. [8]	Huang et al. [22]	Li et al. [10]	proposed
Better stylization	19.2%	8.4%	16.0%	**56.4%**
Fewer artifacts	21.6%	6.0%	6.8%	**65.6%**

decreasing λ helps remove artifacts and hence improves photorealism. However, if λ is too small, the output image tends to be over-smoothed. In order to find the optimal λ, we perform a grid search. We use the similarity between the boundary maps extracted from stylized and original content photos as the criteria since object boundaries should remain the same despite the stylization [46]. We employ the HED method [45] for boundary detection and use two standard boundary detection metrics: ODS and OIS. A higher ODS or OIS score means a stylized photo better preserves the content in the original photo. The average scores over the benchmark dataset are shown on the rightmost of Figure 8. Based on the results, we use $\lambda = 10^{-4}$ in all the experiments.

Alternative smoothing techniques. In Figure 9, we compare our photorealistic smoothing step with two alternative approaches. In the first approach, we use the PhotoWCT-stylized photo as the initial solution for solving the second optimization problem in the method of Luan et al. [9]. The result is shown in Figure 9(b). This approach leads to noticeable artifacts as the road color is distorted. In the second approach, we use the method of Mechrez et al. [25], which refines stylized results by matching the gradients in the output photo to those in the content photo. As shown in Figure 9(c), we find this approach performs well for removing structural distortions on boundaries but does not remove visual artifacts. In contrast, our method (Figure 9(d)) generates more photorealistic results with an efficient closed-form solution.

Run-time. In Table 3, we compare the run-time of the proposed algorithm to that of the state-of-the-art [9]. We note that while our algorithm has a closed-form solution, Luan et al. [9] rely on non-convex optimization. To stylize a photo, Luan et al. [9] solve two non-convex optimization problems sequentially where a solution[4] to the first optimization problem is used as an initial solution to solve the second optimization problem. We report the total run-time required for obtaining the final stylization results. We resize the content images in the benchmark dataset to different sizes and report the average run-time for each image size. The experiment is conducted on a PC with an NVIDIA Titan X

[4] Note the solution is at most local optimal.

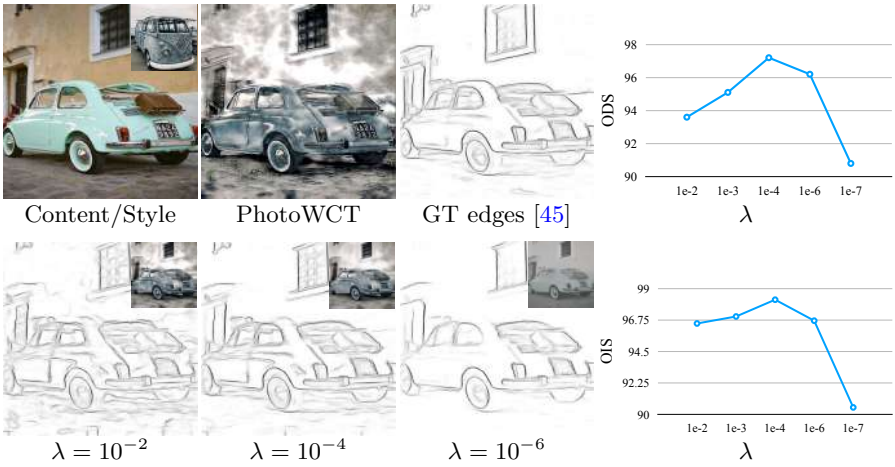


Fig. 8: Visualization of effects of using different λ values in the photorealistic smoothing step. We show the edge maps of different stylization results (inset) at bottom and compare them with the edge map of the content in terms of the ODS and OIS metric (rightmost).

(a) PhotoWCT (b) Luan et al. [9] (c) Mechrez et al. [25] (d) proposed

Fig. 9: Comparison between using our photorealistic smoothing step and other refinement methods (b)-(d).

Pascal GPU. To stylize images of 1024×512 resolution, our algorithm takes 13.16 seconds, which is 49 times faster than 650.45 seconds achieved by Luan et al. [9].

In Table 3, we also report the run-time of each step in our algorithm. We find the smoothing step takes most of the computation time, since it involves inverting the sparse matrix W in (5) using the LU decomposition. By employing efficient LU-decomposition algorithms developed for large sparse matrices, the complexity can be roughly determined by the number of non-zero entries in the matrices only. In our case, since each pixel is only connected to its neighbors (e.g., 3×3 window), the number of non-zero values in W grows linearly with the image size.

For further speed-up, we can approximate the smoothing step using guided image filtering [47], which can smooth the PhotoWCT output based on the content photo. We will refer to this version of our algorithm `approx`. Although approximating the smoothing step with guided image filtering results in slightly degraded performance as comparing to the original algorithm, it leads to a large speed gain as shown in Table 3. To stylize images of 1024×512 resolution, `approx` only takes 0.64 seconds, which is 1,016 times faster than 650.45 seconds achieved by Luan et al. [9]. To quantify the performance degradation due to the approximation, we conduct additional user studies comparing the proposed algorithm and its

Table 3: Run-time comparison. We compute the average run time (in seconds) of the evaluated algorithms across various image resolutions.

Image resolution	Luan et al.[9]	proposed	PhotoWCT	smoothing	`approx`
256×128	79.61	0.96	0.40	0.56	0.41
512×256	186.52	2.95	0.42	2.53	0.47
768×384	380.82	7.05	0.53	6.52	0.55
1024×512	650.45	13.16	0.56	12.60	0.64

Table 4: User preference score comparison: comparing `approx` (the fast approximation of the proposed algorithm) to the proposed algorithm as well as other photorealistic stylization algorithms.

	proposed/`approx`	Luan et al. [9]/`approx`	Pitié et al. [2]/`approx`
Better stylization	**59.6%** / 40.4	36.4 / **63.6%**	46.0 / **54.0%**
Fewer artifacts	**52.8%** / 47.2	20.8 / **79.2%**	46.8 / **53.2%**

Content/Style Reinhard et al. [1] Pitié et al. [2] Luan et al. [9] Ours

Fig. 10: Failure case. Both the proposed and other photorealistic stylization algorithms fail to transfer the flower patterns to the pot.

approximation. We use the same evaluation protocol as described above. The results are shown in Table 4. In general, the stylization results rendered by `approx` are less preferred by the users as compared to those generated by the full algorithm. However, the results from `approx` are still preferred over other methods in terms of both stylization effects and photorealism.

Failure case. Figure 10 shows a failure case where the proposed method fails to transfer the flower patterns in the style photo to the content photo. Similar limitations also apply to the other photorealistic stylization methods [2,1,9]. Since the proposed method uses the pixel affinity of the content in the photorealistic smoothing step, it favors a stylization output with smooth color transition on the pot surface as in the input photo.

5 Conclusions

We presented a novel fast photorealistic image stylization method. It consists of a stylization step and a photorealistic smoothing step. Both steps have efficient closed-form solutions. Experimental results show that our algorithm generates stylization outputs that are much more preferred by human subject as compared to those by the state-of-the-art, while running much faster.

References

1. Reinhard, E., Ashikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Computer graphics and applications **21**(5) (2001) 34–41 1, 2, 8, 9, 10, 14, 18, 19, 20, 21, 22, 23
2. Pitié, F., Kokaram, A.C., Dahyot, R.: N-dimensional probability density function transfer and its application to color transfer. In: ICCV. (2005) 1, 2, 8, 9, 10, 11, 12, 14, 18, 19, 20, 21, 22, 23
3. Sunkavalli, K., Johnson, M.K., Matusik, W., Pfister, H.: Multi-scale image harmonization. ACM Transactions on Graphics **29**(4) (2010) 125 1
4. Bae, S., Paris, S., Durand, F.: Two-scale tone management for photographic look. ACM Transactions on Graphics **25**(3) (2006) 637–645 1
5. Laffont, P.Y., Ren, Z., Tao, X., Qian, C., Hays, J.: Transient attributes for high-level understanding and editing of outdoor scenes. ACM Transactions on Graphics **33**(4) (2014) 149 1, 2
6. Shih, Y., Paris, S., Barnes, C., Freeman, W.T., Durand, F.: Style transfer for headshot portraits. In: SIGGRAPH. (2014) 1, 2
7. Gatys, L.A., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks. In: NIPS. (2015) 1, 3
8. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR. (2016) 1, 2, 3, 8, 10, 12, 18, 19, 20, 21, 22, 23
9. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer. In: CVPR. (2017) 1, 2, 3, 8, 9, 10, 11, 12, 13, 14, 18, 19, 20, 21, 22, 23
10. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Universal style transfer via feature transforms. In: NIPS. (2017) 2, 3, 4, 6, 8, 10, 11, 12, 17, 18, 19, 20, 21, 22, 23
11. Freedman, D., Kisilev, P.: Object-to-object color transfer: Optimal flows and smsp transformations. In: CVPR. (2010) 2
12. Shih, Y., Paris, S., Durand, F., Freeman, W.T.: Data-driven hallucination of different times of day from a single outdoor photo. In: SIGGRAPH. (2013) 2
13. Wu, F., Dong, W., Kong, Y., Mei, X., Paul, J.C., Zhang, X.: Content-based colour transfer. Computer Graphics Forum **32**(1) (2013) 190–203 2
14. Tsai, Y.H., Shen, X., Lin, Z., Sunkavalli, K., Yang, M.H.: Sky is not the limit: Semantic-aware sky replacement. ACM Transactions on Graphics **35**(4) (2016) 149 2
15. Li, C., Wand, M.: Combining markov random fields and convolutional neural networks for image synthesis. In: CVPR. (2016) 3
16. Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V.: Texture networks: Feed-forward synthesis of textures and stylized images. In: ICML. (2016) 3
17. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV. (2016) 3, 8
18. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Diversified texture synthesis with feed-forward networks. In: CVPR. (2017) 3
19. Chen, D., Yuan, L., Liao, J., Yu, N., Hua, G.: Stylebank: An explicit representation for neural image style transfer. In: CVPR. (2017) 3
20. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. In: ICLR. (2017) 3
21. Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., Shlens, J.: Exploring the structure of a real-time, arbitrary neural artistic stylization network. In: BMVC. (2017) 3

22. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: ICCV. (2017) 3, 8, 10, 12, 18, 19, 20, 21, 22, 23
23. Liao, J., Yao, Y., Yuan, L., Hua, G., Kang, S.B.: Visual attribute transfer through deep image analogy. arXiv preprint arXiv:1705.01088 (2017) 3
24. Li, S., Xu, X., Nie, L., Chua, T.S.: Laplacian-steered neural style transfer. In: ACM MM. (2017) 3
25. Mechrez, R., Shechtman, E., Zelnik-Manor, L.: Photorealistic style transfer with screened poisson equation. In: BMVC. (2017) 3, 12, 13
26. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR. (2017) 3
27. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional gans. In: CVPR. (2018) 3
28. Liu, M.Y., Tuzel, O.: Coupled generative adversarial networks. In: NIPS. (2016) 3
29. Taigman, Y., Polyak, A., Wolf, L.: Unsupervised cross-domain image generation. In: ICLR. (2017) 3
30. Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., Webb, R.: Learning from simulated and unsupervised images through adversarial training. In: CVPR. (2017) 3
31. Liu, M.Y., Breuel, T., Kautz, J.: Unsupervised image-to-image translation networks. In: NIPS. (2017) 3
32. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV. (2017) 3
33. Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation. In: ECCV. (2018) 3
34. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR. (2015) 4, 8, 17
35. Zhao, J., Mathieu, M., Goroshin, R., LeCun, Y.: Stacked what-where auto-encoders. In: ICLR Workshop. (2016) 5
36. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: ECCV. (2014) 5
37. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: ICCV. (2015) 5
38. Zhou, D., Weston, J., Gretton, A., Bousquet, O., Schölkopf, B.: Ranking on data manifolds. In: NIPS. (2004) 6
39. Yang, C., Zhang, L., Lu, H., Ruan, X., Yang, M.H.: Saliency detection via graph-based manifold ranking. In: CVPR. (2013) 6
40. Shi, J., Malik, J.: Normalized cuts and image segmentation. PAMI **22**(8) (2000) 888–905 7
41. Levin, A., Lischinski, D., Weiss, Y.: A closed-form solution to natural image matting. PAMI **30**(2) (2008) 228–242 7
42. Zelnik-Manor, L., Perona, P.: Self-tuning spectral clustering. In: NIPS. (2005) 7
43. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV. (2014) 8
44. Gatys, L.A., Ecker, A.S., Bethge, M., Hertzmann, A., Shechtman, E.: Controlling perceptual factors in neural style transfer. In: CVPR. (2017) 8, 18
45. Xie, S., Tu, Z.: Holistically-nested edge detection. In: ICCV. (2015) 12, 13
46. Cutzu, F., Hammoud, R., Leykin, A.: Estimating the photorealism of images: Distinguishing paintings from photographs. In: CVPR. (2003) 12
47. He, K., Sun, J., Tang, X.: Guided image filtering. PAMI **35**(6) (2013) 1397–1409 13

A Multi-level Stylization

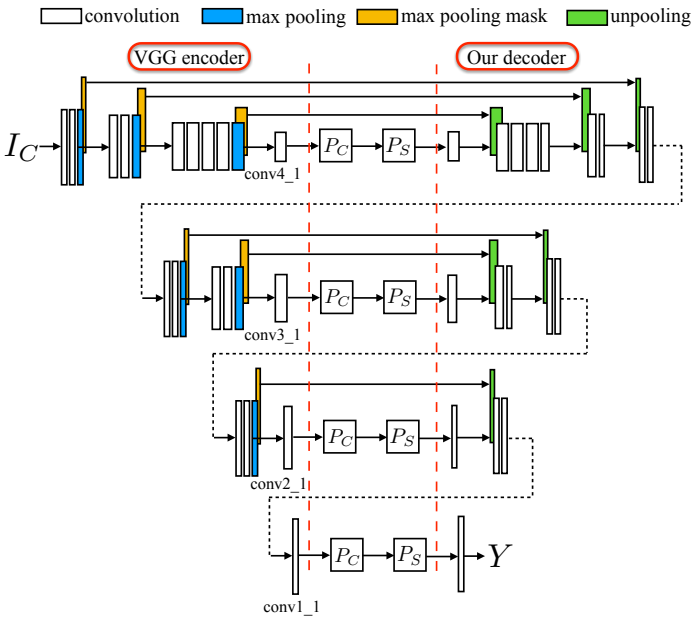


Fig. 11: Illustration of the multi-level stylization scheme

The PhotoWCT stylization step utilizes an auto-encoder with unpooling layers and a pair of feature transforms (P_C, P_S). The encoder is made of the first few layers of the VGG-19 [34] network. The feature transforms are applied to the features extracted by the encoder. As suggested in the WCT [10], we match features across different levels in the VGG-19 encoder to fully capture the characteristics of the style. Specifically, we train four decoders for image reconstruction. They are responsible for inverting features extracted from *conv*1_1, *conv*2_1, *conv*3_1, and *conv*4_1 layer of VGG-19, respectively. With the four encoders, we have a set of 4 auto-encoder networks, which corresponds to a set of 4 PhotoWCT transforms. We first apply the transform that uses the deepest feature representation to stylize the content image. The stylized image is then passed to the transform that uses the second highest feature representation as shown in Figure 11. Note that the decoders are trained separately and they do not share weights.

B Network Architecture

Table 5 shows the detailed configurations of the decoders. We use the following abbreviation for ease of presentation: N=Filter number, K=Filter size, S=Stride.

Table 5: Details of the decoders.

Layer Name	Specification	Decoder 1	Decoder 2	Decoder 3	Decoder 4
$inv - conv4\_1$	Conv (N256, K3, S1), ReLU	v			
	MaxUnpooling (K2, S2)	v			
$inv - conv3\_4$	Conv (N256, K3, S1), ReLU	v			
$inv - conv3\_3$	Conv (N256, K3, S1), ReLU	v			
$inv - conv3\_2$	Conv (N256, K3, S1), ReLU	v			
$inv - conv3\_1$	Conv (N128, K3, S1), ReLU	v	v		
	MaxUnpooling (K2, S2)	v	v		
$inv - conv2\_2$	Conv (N128, K3, S1), ReLU	v	v		
$inv - conv2\_1$	Conv (N64, K3, S1), ReLU	v	v	v	
	MaxUnpooling (K2, S2)	v	v	v	
$inv - conv1\_2$	Conv (N64, K3, S1), ReLU	v	v	v	
$inv - conv1\_1$	Conv (N3, K3, S1)	v	v	v	v

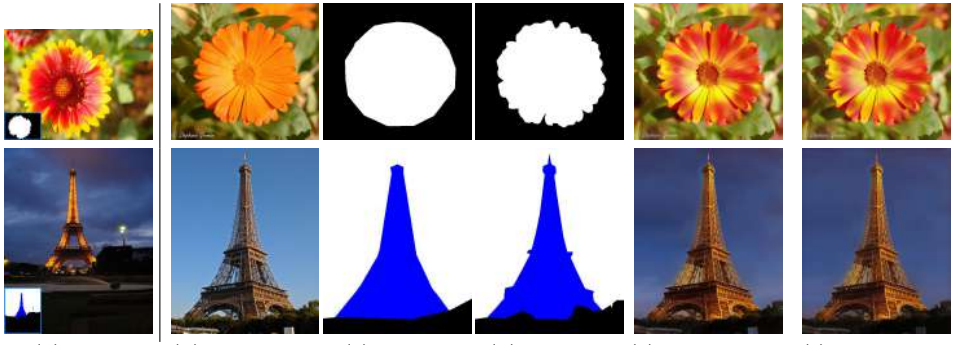
C Semantic Label Map

The proposed algorithm can leverage semantic label maps for better content–style matching when they are available, similar to the prior work [44,9]. We only use the label map for finding matching areas between content and style images. The specific class information is not used. We further note that the proposed algorithm does not require the label map to be drawn precisely along object boundaries. The photorealistic smoothing step, which employs pixel affinities to encourage consistent stylization, could accommodate imprecise boundary annotations. This could greatly reduce the labeling burden for users. Figure 12 shows the comparisons between using coarse and precise label maps. The results in (e) and (f) show that using the coarse map can achieve nearly the same stylization performance as using the precise map.

D Additional Results

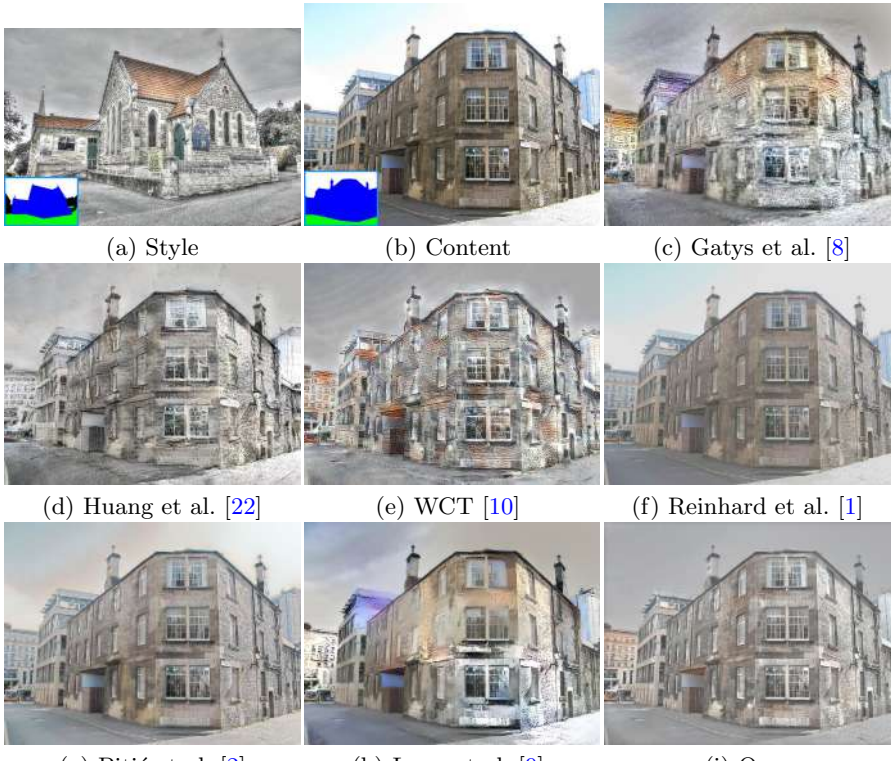
We show more photorealistic stylization results in Figure 13 to Figure 17. In each figure, we first present the content–style pair together with their corresponding label maps in (a) and (b). The maps are either from Luan et al. [9] or roughly drawn by human. Each color represents a different semantic label.

We compare our method with three artistic stylization methods [8,22,10] in (c)–(e) and three photorealistic stylization methods [1,2,9] in (f)–(h). The results show that our method generates more photorealistic results with much less structural artifacts and more consistent stylizations for a variety of examples.



(a) Style (b) Content (c) Coarse map (d) Precise map (e) Stylization w/ coarse map (f) Stylization w/ precise map

Fig. 12: Comparisons of stylization results between drawing the coarse and precise label maps in the content.

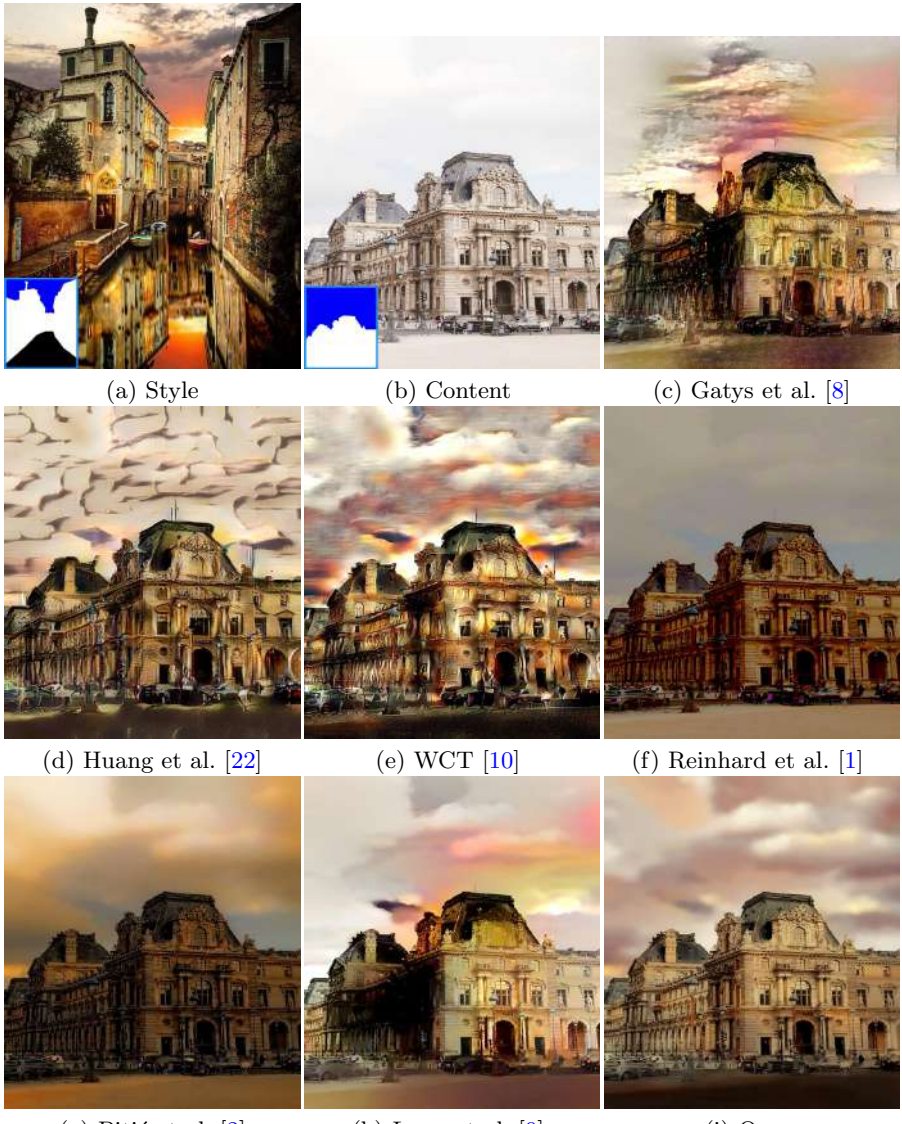


(a) Style (b) Content (c) Gatys et al. [8]

(d) Huang et al. [22] (e) WCT [10] (f) Reinhard et al. [1]

(g) Pitié et al. [2] (h) Luan et al. [9] (i) Ours

Fig. 13: Comparisons of different stylization methods.

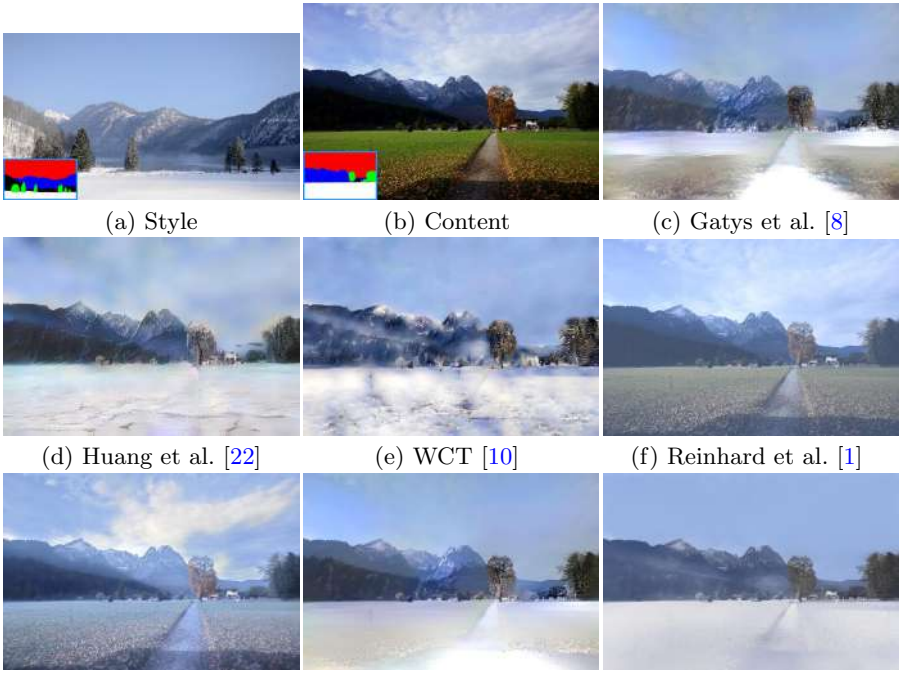


(a) Style (b) Content (c) Gatys et al. [8]

(d) Huang et al. [22] (e) WCT [10] (f) Reinhard et al. [1]

(g) Pitié et al. [2] (h) Luan et al. [9] (i) Ours

Fig. 14: Comparisons of different stylization methods.

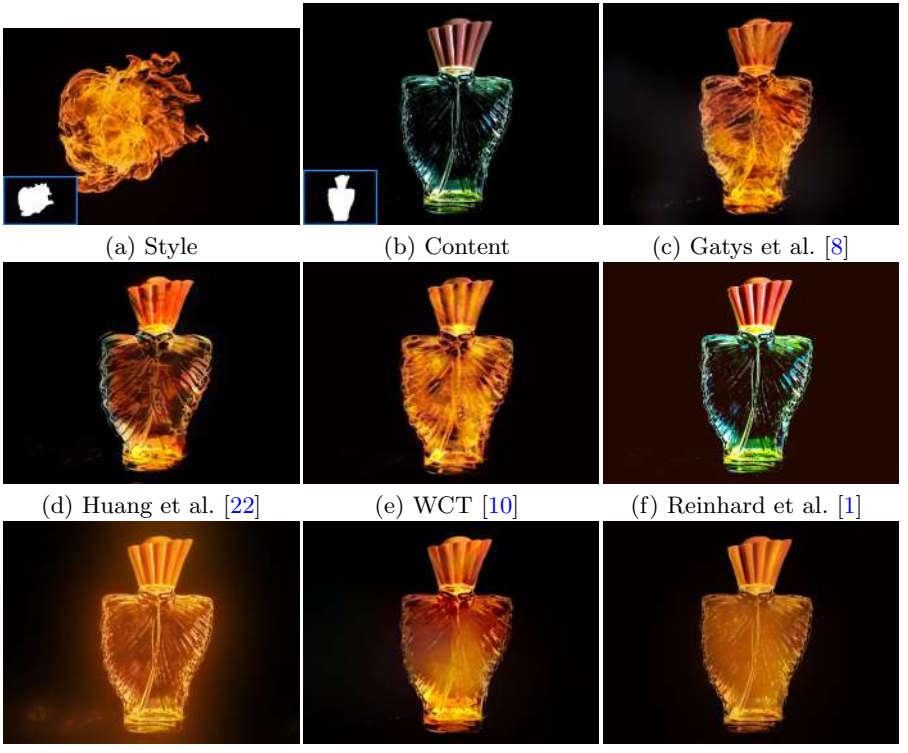


(a) Style (b) Content (c) Gatys et al. [8]

(d) Huang et al. [22] (e) WCT [10] (f) Reinhard et al. [1]

(g) Pitié et al. [2] (h) Luan et al. [9] (i) Ours

Fig. 15: Comparisons of different stylization methods.



(a) Style (b) Content (c) Gatys et al. [8]

(d) Huang et al. [22] (e) WCT [10] (f) Reinhard et al. [1]

(g) Pitié et al. [2] (h) Luan et al. [9] (i) Ours

Fig. 16: Comparisons of different stylization methods.

(a) Style (b) Content (c) Gatys et al. [8]

(d) Huang et al. [22] (e) WCT [10] (f) Reinhard et al. [1]

(g) Pitié et al. [2] (h) Luan et al. [9] (i) Ours

Fig. 17: Comparisons of different stylization methods.

Taskonomy: Disentangling Task Transfer Learning

Amir R. Zamir[1,2] Alexander Sax[1*] William Shen[1*] Leonidas Guibas[1] Jitendra Malik[2] Silvio Savarese[1]

[1] Stanford University [2] University of California, Berkeley

http://taskonomy.vision/

Abstract

*Do visual tasks have a relationship, or are they unrelated? For instance, could having surface normals simplify estimating the depth of an image? Intuition answers these questions positively, implying existence of a **structure** among visual tasks. Knowing this structure has notable values; it is the concept underlying transfer learning and provides a principled way for identifying redundancies across tasks, e.g., to seamlessly reuse supervision among related tasks or solve many tasks in one system without piling up the complexity.*

We proposes a fully computational approach for modeling the structure of space of visual tasks. This is done via finding (first and higher-order) transfer learning dependencies across a dictionary of twenty six 2D, 2.5D, 3D, and semantic tasks in a latent space. The product is a computational taxonomic map for task transfer learning. We study the consequences of this structure, e.g. nontrivial emerged relationships, and exploit them to reduce the demand for labeled data. For example, we show that the total number of labeled datapoints needed for solving a set of 10 tasks can be reduced by roughly $\frac{2}{3}$ (compared to training independently) while keeping the performance nearly the same. We provide a set of tools for computing and probing this taxonomical structure including a solver that users can employ to devise efficient supervision policies for their use cases.

1. Introduction

Object recognition, depth estimation, edge detection, pose estimation, etc are examples of common vision tasks deemed useful and tackled by the research community. Some of them have rather clear relationships: we understand that surface normals and depth are related (one is a derivate of the other), or vanishing points in a room are useful for orientation. Other relationships are less clear: how keypoint detection and the shading in a room can, together, perform pose estimation.

*Equal.

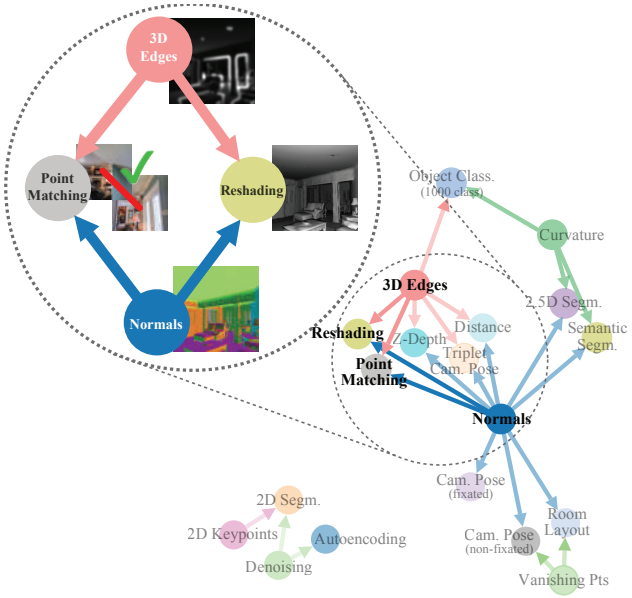


Figure 1: **A sample task structure discovered by the computational task taxonomy (*taskonomy*)**. It found that, for instance, by combining the learned features of a surface normal estimator and occlusion edge detector, good networks for reshading and point matching can be rapidly trained with little labeled data.

The field of computer vision has indeed gone far without explicitly using these relationships. We have made remarkable progress by developing advanced learning machinery (e.g. ConvNets) capable of finding complex mappings from X to Y when many pairs of (x, y) s.t. $x \in X, y \in Y$ are given as training data. This is usually referred to as fully supervised learning and often leads to problems being solved in isolation. Siloing tasks makes training a new task or a comprehensive perception system a Sisyphean challenge, whereby each task needs to be learned individually from scratch. Doing so ignores their quantifiably useful relationships leading to a massive labeled data requirement.

Alternatively, a model aware of the relationships among tasks demands less supervision, uses less computation, and behaves in more predictable ways. Incorporating such a structure is the first stepping stone towards develop-

ing provably efficient comprehensive/universal perception models [34, 4], i.e. ones that can solve a large set of tasks before becoming intractable in supervision or computation demands. However, this task space structure and its effects are still largely unknown. The relationships are non-trivial, and finding them is complicated by the fact that we have imperfect learning models and optimizers. In this paper, we attempt to shed light on this underlying structure and present a framework for mapping the space of visual tasks. Here what we mean by "structure" is a collection of computationally found relations specifying which tasks supply useful information to another, and by how much (see Fig. 1).

We employ a fully computational approach for this purpose, with neural networks as the adopted computational function class. In a feedforward network, each layer successively forms more abstract representations of the input containing the information needed for mapping the input to the output. These representations, however, can transmit statistics useful for solving other outputs (tasks), presumably if the tasks are related in some form [83, 19, 58, 46]. This is the basis of our approach: we computes an affinity matrix among tasks based on whether the solution for one task can be sufficiently easily read out of the representation trained for another task. Such transfers are exhaustively sampled, and a Binary Integer Programming formulation extracts a globally efficient transfer policy from them. We show this model leads to solving tasks with far less data than learning them independently and the resulting structure holds on common datasets (ImageNet [78] and Places [104]).

Being fully computational and representation-based, the proposed approach avoids imposing prior (possibly incorrect) assumptions on the task space. This is crucial because the priors about task relations are often derived from either human intuition or analytical knowledge, while neural networks need not operate on the same principles [63, 33, 40, 45, 102, 88]. For instance, although we might expect depth to transfer to surface normals better (derivatives are easy), the opposite is found to be the better direction in a computational framework (i.e. suited neural networks better).

An interactive taxonomy solver which uses our model to suggest data-efficient curricula, a live demo, dataset, and code are available at http://taskonomy.vision/.

2. Related Work

Assertions of existence of a structure among tasks date back to the early years of modern computer science, e.g. with Turing arguing for using learning elements [95, 98] rather than the final outcome or Jean Piaget's works on developmental stages using previously learned stages as sources [74, 39, 38], and have extended to recent works [76, 73, 50, 18, 97, 61, 11, 66]. Here we make an attempt to actually find this structure. We acknowledge that this is related to a breadth of topics, e.g. compositional modeling [35, 10, 13, 23, 55, 92, 90], homomorphic cryptography [42], lifelong learning [93, 15, 85, 84], functional maps [71], certain aspects of Bayesian inference and Dirichlet processes [54, 91, 90, 89, 37, 39], few-shot learning [81, 25, 24, 70, 86], transfer learning [75, 84, 29, 64, 67, 59], un/semi/self-supervised learning [22, 8, 17, 103, 19, 83], which are studied across various fields [73, 94, 12]. We review the topics most pertinent to vision within the constraints of space:

Self-supervised learning methods leverage the inherent relationships between tasks to learn a desired expensive one (e.g. object detection) via a cheap surrogate (e.g. colorization) [68, 72, 17, 103, 100, 69]. Specifically, they use a manually-entered local part of the structure in the task space (as the surrogate task is manually defined). In contrast, our approach models this large space of tasks in a computational manner and can discover obscure relationships.

Unsupervised learning is concerned with the redundancies in the input domain and leveraging them for forming compact representations, which are usually agnostic to the downstream task [8, 49, 20, 9, 32, 77]. Our approach is not unsupervised by definition as it is not agnostic to the tasks. Instead, it models the space tasks belong to and in a way utilizes the *functional* redundancies among tasks.

Meta-learning generally seeks performing the learning at a level higher than where conventional learning occurs, e.g. as employed in reinforcement learning [21, 31, 28], optimization [2, 82, 48], or certain architectural mechanisms [27, 30, 87, 65]. The motivation behind meta learning has similarities to ours and our outcome can be seen as a computational meta-structure of the space of tasks.

Multi-task learning targets developing systems that can provide multiple outputs for an input in one run [50, 18]. Multi-task learning has experienced recent progress and the reported advantages are another support for existence of a useful structure among tasks [93, 100, 50, 76, 73, 50, 18, 97, 61, 11, 66]. Unlike multi-task learning, we explicitly model the relations among tasks and extract a meta-structure. The large number of tasks we consider also makes developing one multi-task network for all infeasible.

Domain adaption seeks to render a function that is developed on a certain domain applicable to another [44, 99, 5, 80, 52, 26, 36]. It often addresses a shift in the *input* domain, e.g. webcam images to D-SLR [47], while the task is kept the same. In contrast, our framework is concerned with *output* (task) space, hence can be viewed as *task/output adaptation*. We also perform the adaptation in a larger space among many elements, rather than two or a few.

In the context of our approach to modeling transfer learning across tasks:

Learning Theoretic approaches may overlap with any of the above topics and usually focus on providing generalization guarantees. They vary in their approach: e.g. by modeling transferability with the transfer family required

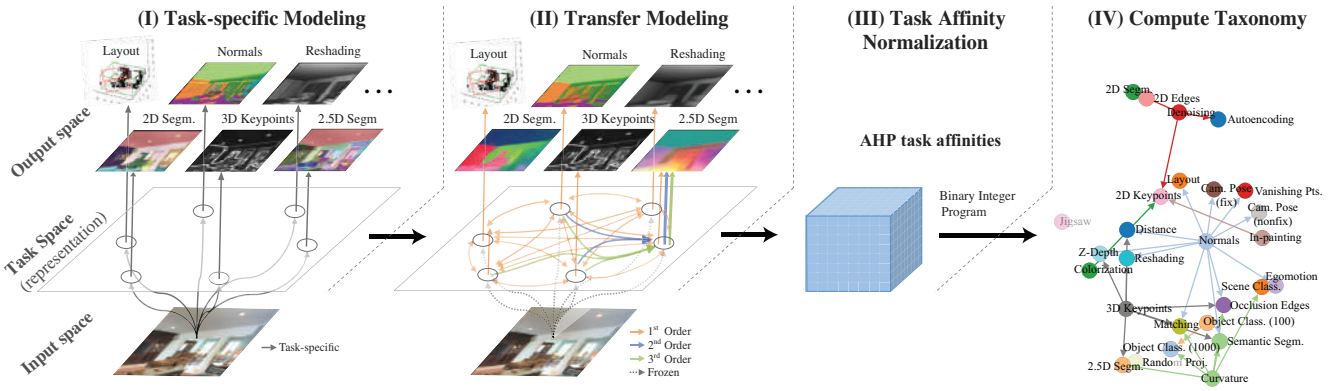


Figure 2: **Computational modeling of task relations and creating the taxonomy.** From left to right: I. Train task-specific networks. II. Train (first order and higher) transfer functions among tasks in a latent space. III. Get normalized transfer affinities using AHP (Analytic Hierarchy Process). IV. Find global transfer taxonomy using BIP (Binary Integer Program).

to map a hypothesis for one task onto a hypothesis for another [7], through information-based approaches [60], or through modeling inductive bias [6]. For these guarantees, learning theoretic approaches usually rely on intractable computations, or avoid such computations by restricting the model or task. Our method draws inspiration from theoretical approaches but eschews (for now) theoretical guarantees in order to use modern neural machinery.

3. Method

We define the problem as follows: we want to maximize the collective performance on a set of tasks $\mathcal{T} = \{t_1, ..., t_n\}$, subject to the constraint that we have a limited supervision budget γ (due to financial, computational, or time constraints). We define our supervision budget γ to be the maximum allowable number of tasks that we are willing to train from scratch (i.e. *source* tasks). The task dictionary is defined as $\mathcal{V}$=$\mathcal{T} \cup \mathcal{S}$ where $\mathcal{T}$ is the set of tasks which we want solved (*target*), and $\mathcal{S}$ is the set of tasks that can be trained (*source*). Therefore, $\mathcal{T} - \mathcal{T} \cap \mathcal{S}$ are the tasks that we want solved but cannot train ("target-only"), $\mathcal{T} \cap \mathcal{S}$ are the tasks that we want solved but could play as source too, and $\mathcal{S} - \mathcal{T} \cap \mathcal{S}$ are the "source-only" tasks which we may not directly care about to solve (e.g. jigsaw puzzle) but can be optionally used if they increase the performance on $\mathcal{T}$.

The **task** tax**onomy** (***taskonomy***) is a computationally found directed hypergraph that captures the notion of task transferability over any given task dictionary. An edge between a group of source tasks and a target task represents a feasible transfer case and its weight is the prediction of its performance. We use these edges to estimate the globally optimal transfer policy to solve $\mathcal{T}$. Taxonomy produces a family of such graphs, parameterized by the available supervision budget, chosen tasks, transfer orders, and transfer functions' expressiveness.

Taxonomy is built using a four step process depicted in Fig. 2. In stage I, a task-specific network for each task in $\mathcal{S}$ is trained. In stage II, all feasible transfers between sources and targets are trained. We include higher-order transfers which use multiple inputs task to transfer to one target. In stage III, the task affinities acquired from transfer function performances are normalized, and in stage IV, we synthesize a hypergraph which can predict the performance of any transfer policy and optimize for the optimal one.

A vision task is an abstraction read from a raw image. We denote a task t more formally as a function f_t which maps image I to $f_t(I)$. Our dataset, $\mathcal{D}$, contains for each task t a set of training pairs $(I, f_t(I))$, e.g. $(image, depth)$.

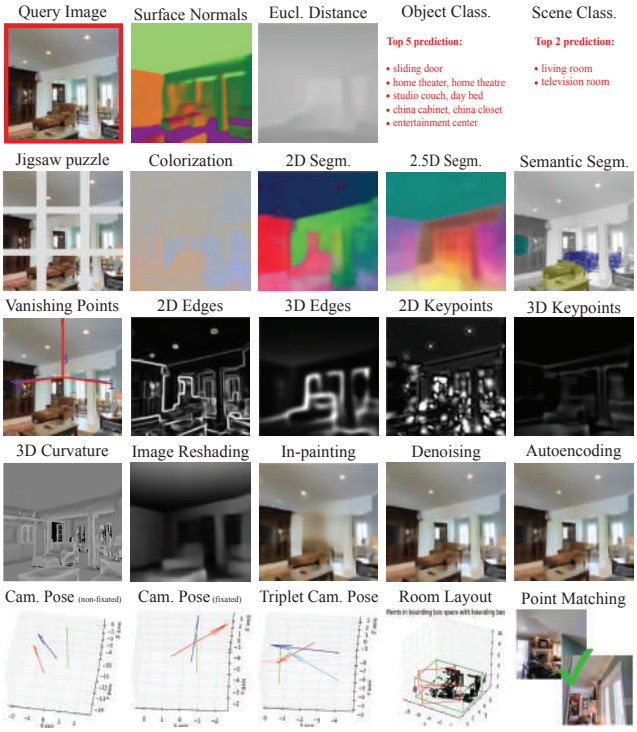


Figure 3: **Task Dictionary.** Outputs of 24 (of 26) task-specific networks for a query (top left). See results of applying frame-wise on a video here.

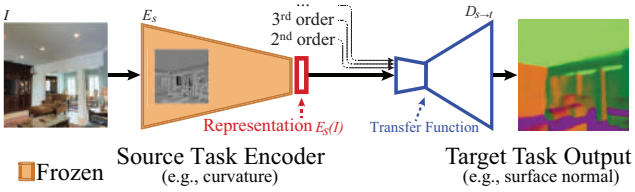


Figure 4: **Transfer Function**. A small readout function is trained to map representations of source task's frozen encoder to target task's labels. If order>1, transfer function receives representations from multiple sources.

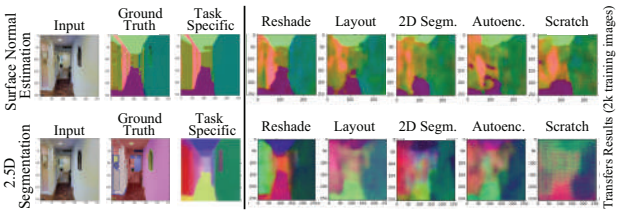


Figure 5: **Transfer results** to normals (upper) and 2.5D Segmentation (lower) from 5 different source tasks. The spread in transferability among different sources is apparent, with reshading among top-performing ones in this case. Task-specific networks were trained on 60x more data. "Scratch" was trained from scratch without transfer learning.

Task Dictionary: Our mapping of task space is done via (26) tasks included in the dictionary, so we ensure they cover common themes in computer vision (2D, 3D, semantics, etc) to the elucidate fine-grained structures of task space. See Fig. 3 for some of the tasks with detailed definition of each task provided in the supplementary material. We include tasks with various levels of abstraction, ranging from solvable by a simple kernel convolved over the image (e.g. edge detection) to tasks requiring basic understanding of scene geometry (e.g. vanishing points) and more abstract ones involving semantics (e.g. scene classification).

It is critical to note the task dictionary is meant to be a *sampled set, not an exhaustive list*, from a denser space of all conceivable visual tasks. Sampling gives us a tractable way to sparsely model a dense space, and the hypothesis is that (subject to a proper sampling) the derived model should generalize to out-of-dictionary tasks. The more regular / better sampled the space, the better the generalization. We evaluate this in Sec. 4.2 with supportive results. For evaluation of the robustness of results w.r.t the choice of dictionary, see the supplementary material.

Dataset: We need a dataset that has annotations for *every task on every image*. Training all of our tasks on exactly the same pixels eliminates the possibility that the observed transferabilities are affected by different input data peculiarities rather than only task intrinsics. There has not been such a dataset of scale made of real images, so we created a dataset of 4 million images of indoor scenes from about 600 buildings; every image has an annotation for every task. The images are registered on and aligned with building-wide meshes similar to [3, 101, 14] enabling us to programmatically compute the ground truth for many tasks without human labeling. For the tasks that still require labels (e.g. scene classes), we generate them using Knowledge Distillation [43] from known methods [104, 57, 56, 78]. See the supplementary material for full details of the process and a user study on the final quality of labels generated using Knowledge Distillation (showing $< 7\%$ error).

3.1. Step I: Task-Specific Modeling

We train a fully supervised task-specific network for each task in $\mathcal{S}$. Task-specific networks have an encoder-decoder architecture homogeneous across all tasks, where the encoder is large enough to extract powerful representations, and the decoder is large enough to achieve a good performance but is much smaller than the encoder.

3.2. Step II: Transfer Modeling

Given a source task s and a target task t, where $s \in \mathcal{S}$ and $t \in \mathcal{T}$, a transfer network learns a small readout function for t given a statistic computed for s (see Fig 4). The statistic is the representation for image I from the encoder of s: $E_s(I)$. The readout function ($D_{s\to t}$) is parameterized by $\theta_{s\to t}$ minimizing the loss L_t:

$$D_{s\to t} := \arg\min_{\theta} \mathbb{E}_{I\in\mathcal{D}}\Big[L_t\Big(D_\theta\big(E_s(I)\big), f_t(I)\Big)\Big], \quad (1)$$

where $f_t(I)$ is ground truth of t for image I. $E_s(I)$ may or may not be sufficient for solving t depending on the relation between t and s (examples in Fig. 5). Thus, the performance of $D_{s\to t}$ is a useful metric as task affinity. We train transfer functions for all feasible source-target combinations.

Accessibility: For a transfer to be successful, the latent representation of the source should both be *inclusive* of sufficient information for solving the target and have the information *accessible*, i.e. easily extractable (otherwise, the raw image or its compression based representations would be optimal). Thus, it is crucial for us to adopt a low-capacity (small) architecture as transfer function trained with a small amount of data, in order to measure transferability conditioned on being highly accessible. We use a shallow fully convolutional network and train it with little data (8x to 120x less than task-specific networks).

Higher-Order Transfers: Multiple source tasks can contain complementary information for solving a target task (see examples in Fig 6). We include higher-order transfers which are the same as first order but receive multiple representations in the input. Thus, our transfers are functions $D : \wp(\mathcal{S}) \to \mathcal{T}$, where $\wp$ is the powerset operator.

As there is a combinatorial explosion in the number of feasible higher-order transfers ($|\mathcal{T}| \times \binom{|\mathcal{S}|}{k}$ for k^{th} order), we employ a sampling procedure with the goal of filtering out higher-order transfers that are less likely to yield good results, without training them. We use a beam search: for

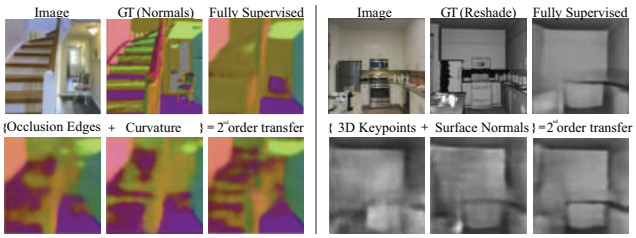


Figure 6: **Higher-Order Transfers.** Representations can contain complementary information. E.g. by transferring simultaneously from 3D Edges and Curvature individual stairs were brought out. See our publicly available interactive transfer visualization page for more examples.

transfers of order $k \leq 5$ to a target, we select its 5 best sources (according to 1^{st} order performances) and include all of their order-k combination. For $k \geq 5$, we use a beam of size 1 and compute the transfer from the top k sources.

Transitive Transfers: We examined if transitive task transfers ($s \rightarrow t_1 \rightarrow t_2$) could improve the performance over their direct counterpart ($a \rightarrow t_2$), but found that the two had equal performance in almost all cases in both high-data and low-data scenarios. The experiment is provided in the supplementary material. Therefore, we need not consider the cases where branching would be more than one level deep when searching for the optimal transfer path.

3.3. Step III: Ordinal Normalization using Analytic Hierarchy Process (AHP)

We want to have an affinity matrix of transferabilities across tasks. Aggregating the raw losses/evaluations $L_{s \rightarrow t}$ from transfer functions into a matrix is obviously problematic as they have vastly different scales and live in different spaces (see Fig. 7-left). Hence, a proper normalization is needed. A naive solution would be to linearly rescale each row of the matrix to the range $[0, 1]$. This approach fails when the actual output quality increases at different speeds w.r.t. the loss. As the loss-quality curve is generally unknown, such approaches to normalization are ineffective.

Instead, we use an *ordinal* approach in which the output quality and loss are only assumed to change monotonically. For each t, we construct W_t a pairwise tournament matrix between all feasible sources for transferring to t. The element at (i, j) is the percentage of images in a held-out test set, $\mathcal{D}_{test}$, on which s_i transfered to t better than s_j did (i.e. $D_{s_i \rightarrow t}(I) > D_{s_j \rightarrow t}(I)$).

We clip this intermediate pairwise matrix W_t to be in $[0.001, 0.999]$ as a form of Laplace smoothing. Then we divide $W'_t = W_t / W_t^T$ so that the matrix shows how many times better s_i is compared to s_j. The final tournament ratio matrix is positive reciprocal with each element $w'_{i,j}$ of W'_t:

$$w'_{i,j} = \frac{\mathbb{E}_{I \in \mathcal{D}_{test}}[D_{s_i \rightarrow t}(I) > D_{s_j \rightarrow t}(I)]}{\mathbb{E}_{I \in \mathcal{D}_{test}}[D_{s_i \rightarrow t}(I) < D_{s_j \rightarrow t}(I)]}. \quad (2)$$

We quantify the final transferability of s_i to t as the corresponding (i^{th}) component of the principal eigenvector of W'_t (normalized to sum to 1). The elements of the principal eigenvector are a measure of centrality, and are proportional to the amount of time that an infinite-length random walk on W'_t will spend at any given source [62]. We stack the principal eigenvectors of W'_t for all $t \in \mathcal{T}$, to get an affinity matrix P ('p' for performance)—see Fig. 7, right.

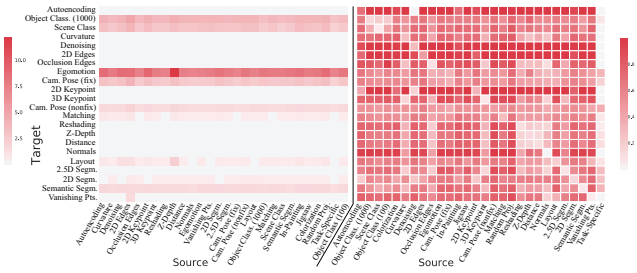


Figure 7: **First-order task affinity matrix** before (left) and after (right) Analytic Hierarchy Process (AHP) normalization. Lower means better transfered. For visualization, we use standard affinity-distance method $dist = e^{-\beta \cdot P}$ (where $\beta = 20$ and e is element-wise matrix exponential). See supplementary material for the full matrix with higher-order transfers.

This approach is derived from Analytic Hierarchy Process [79], a method widely used in operations research to create a total order based on multiple pairwise comparisons.

3.4. Step IV: Computing the Global Taxonomy

Given the normalized task affinity matrix, we need to devise a global transfer policy which maximizes collective performance across all tasks, while minimizing the used supervision. This problem can be formulated as subgraph selection where tasks are nodes and transfers are edges. The optimal subgraph picks the ideal source nodes and the best edges from these sources to targets while satisfying that the number of source nodes does not exceed the supervision budget. We solve this subgraph selection problem using Boolean Integer Programming (BIP), described below, which can be solved optimally and efficiently [41, 16].

Our transfers (edges), E, are indexed by i with the form $(\{s^i_1, \ldots, s^i_{m_i}\}, t^i)$ where $\{s^i_1, \ldots, s^i_{m_i}\} \subset \mathcal{S}$ and $t^i \in \mathcal{T}$. We define operators returning target and sources of an edge:

$$\left(\{s^i_1, \ldots, s^i_{m_i}\}, t^i\right) \xmapsto{sources} \{s^i_1, \ldots, s^i_{m_i}\}$$
$$\left(\{s^i_1, \ldots, s^i_{m_i}\}, t^i\right) \xmapsto{target} t^i.$$

Solving a task t by fully supervising it is denoted as $\left(\{t\}, t\right)$. We also index the targets $\mathcal{T}$ with j so that in this section, i is an edge and j is a target.

The parameters of the problem are: the supervision budget (γ) and a measure of performance on a target from each of its transfers (p_i), i.e. the affinities from P. We can also optionally include additional parameters of: r_j specifying the relative importance of each target task and ℓ_i specifying the relative cost of acquiring *labels* for each task.

The BIP is parameterized by a vector x where each transfer and each task is represented by a binary variable; x indicates which nodes are picked to be source and which transfers are selected. The canonical form for a BIP is:

$$\begin{aligned} &\text{maximize } c^T x\,, \\ &\text{subject to } Ax \preceq b \\ &\text{and } x \in \{0,1\}^{|E|+|\mathcal{V}|}\,. \end{aligned}$$

Each element c_i for a transfer is the product of the importance of its target task and its transfer performance:

$$c_i := r_{target(i)} \cdot p_i\,. \tag{3}$$

Hence, the *collective* performance on all targets is the summation of their individual AHP performance, p_i, weighted by the user specified importance, r_i.

Now we add three types of constraints via matrix A to enforce each feasible solution of the BIP instance corresponds to a valid subgraph for our transfer learning problem: *Constraint I:* if a transfer is included in the subgraph, all of its source nodes/tasks must be included too, *Constraint II:* each target task has exactly one transfer in, *Constraint III:* supervision budget is not exceeded.

Constraint I: For each row a_i in A we require $a_i \cdot x \leq b_i$, where

$$a_{i,k} = \begin{cases} |sources(i)| & \text{if } k = i \\ -1 & \text{if } (k - |E|) \in sources(i) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$b_i = 0. \tag{5}$$

Constraint II: Via the row $a_{|E|+j}$, we enforce that each target has exactly one transfer:

$$a_{|E|+j,i} := 2 \cdot \mathbb{1}_{\{target(i)=j\}}, \qquad b_{|E|+j} := -1. \tag{6}$$

Constraint III: the solution is enforced to not exceed the budget. Each transfer i is assigned a label cost ℓ_i, so

$$a_{|E|+|\mathcal{V}|+1,i} := \ell_i, \qquad b_{|E|+|\mathcal{V}|+1} := \gamma. \tag{7}$$

The elements of A not defined above are set to 0. The problem is now a valid BIP and can be optimally solved in a fraction of a second [41]. The BIP solution $\hat{x}$ corresponds to the optimal subgraph, which is our taxonomy.

4. Experiments

With 26 tasks in the dictionary (4 source-only tasks), our approach leads to training 26 fully supervised task-specific networks, 22×25 transfer networks in 1^{st} order, and $22 \times \binom{25}{k}$ for k^{th} order, from which we sample according to the procedure in Sec. 3. The total number of transfer functions trained for the taxonomy was ∼3,000 which took 47,886 GPU hours on the cloud.

Out of 26 tasks, we usually use the following 4 as source-only tasks (described in Sec. 3) in the experiments: colorization, jigsaw puzzle, in-painting, random projection. However, the method is applicable to an arbitrary partitioning of the dictionary into $\mathcal{T}$ and $\mathcal{S}$. The interactive solver website allows the user to specify any desired partition.

Task	*avg*	*rand*	Task	*avg*	*rand*	Task	*avg*	*rand*
Denoising	100	99.9	Layout	99.6	89.1	Scene Class.	97.0	93.4
Autoenc.	100	99.8	2D Edges	100	99.9	Occ. Edges	100	95.4
Reshading	94.9	95.2	Pose (fix)	76.3	79.5	Pose (nonfix)	60.2	61.9
Inpainting	99.9	-	2D Segm.	97.7	95.7	2.5D Segm.	94.2	89.4
Curvature	78.7	93.4	Matching	86.8	84.6	Egomotion	67.5	72.3
Normals	99.4	99.5	Vanishing	99.5	96.4	2D Keypnt.	99.8	99.4
Z-Depth	92.3	91.1	Distance	92.4	92.1	3D Keypnt.	96.0	96.9
Mean	92.4	90.9						

Table 1: **Task-Specific Networks' Sanity:** Win rates vs. *random* (Gaussian) network representation readout and statistically informed guess *avg*.

Network Architectures: We preserved the architectural and training details across tasks as homogeneously as possible to avoid injecting any bias. The **encoder** architecture is identical across all task-specific networks and is a fully convolutional ResNet-50 without pooling. All **transfer** functions include identical shallow networks with 2 conv layers (concatenated channel-wise if higher-order). The loss (L_t) and **decoder**'s architecture, though, have to depend on the task as the output structures of different tasks vary; for all pixel-to-pixel tasks, e.g. normal estimation, the decoder is a 15-layer fully convolutional network; for low dimensional tasks, e.g. vanishing points, it consists of 2-3 FC layers. All networks are trained using the same hyperparameters regardless of task and on exactly the same input images. Tasks with more than one input, e.g. relative camera pose, share weights between the encoder towers. Transfer networks are all trained using the same hyperparameters as the task-specific networks, except that we anneal the learning rate earlier since they train much faster. Detailed definitions of architectures, training process, and experiments with different encoders can be found in the supplementary material.

Data Splits: Our dataset includes 4 million images. We made publicly available the models trained on full dataset, but for the experiments reported in the main paper, we used a subset of the dataset as the extracted structure stabilized and did not change when using more data (explained in Sec. 5.2). The used subset is partitioned into training (120k), validation (16k), and test (17k) images, each from non-overlapping sets of buildings. Our task-specific networks are trained on the training set and the transfer networks are trained on a subset of validation set, ranging from 1k images to 16k, in order to model the transfer patterns under different data regimes. In the main paper, we report all results under the 16k transfer supervision regime (∼10% of the split) and defer the additional sizes to the supplementary material and website (see Sec. 5.2). Transfer functions are evaluated on the test set.

How good are the trained task-specific networks? *Win rate (%)* is the proportion of test set images for which a baseline is beaten. Table 1 provides win rates of the task-specifc networks vs. two baselines. Visual outputs for a ran-

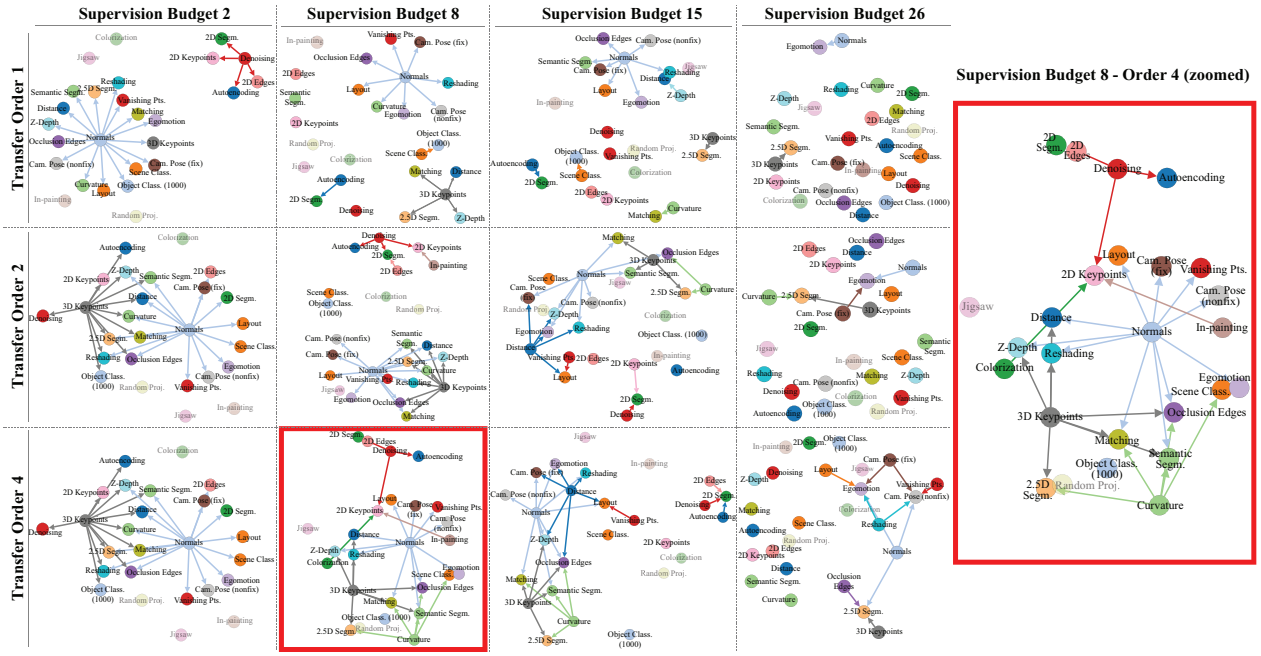


Figure 8: **Computed taxonomies** for solving 22 tasks given various supervision budgets (x-axes), and maximum allowed transfer orders (y-axes). One is magnified for better visibility. Nodes with incoming edges are target tasks, and the number of their incoming edges is the order of their chosen transfer function. Still transferring to some targets when tge budget is 26 (full budget) means certain transfers started performing better than their fully supervised task-specific counterpart. See the interactive solver website for color coding of the nodes by *Gain* and *Quality* metrics. Dimmed nodes are the source-only tasks, and thus, only participate in the taxonomy if found worthwhile by the BIP optimization to be one of the sources.

dom test sample are in Fig. 3. The high win rates in Table 1 and qualitative results show the networks are well trained and stable and can be relied upon for modeling the task space. See results of applying the networks on a YouTube video frame-by-frame here. A live demo for user uploaded queries is available here.

To get a sense of the quality of our networks vs. state-of-the-art task-specific methods, we compared our depth estimator vs. released models of [53] which led to outperforming [53] with a win rate of 88% and losses of 0.35 vs. 0.47 (further details in the supplementary material). In general, we found the task-specific networks to perform on par or better than state-of-the-art for many of the tasks, though we do not formally benchmark or claim this.

4.1. Evaluation of Computed Taxonomies

Fig. 8 shows the computed taxonomies optimized to solve the full dictionary, i.e. all tasks are placed in $\mathcal{T}$ and $\mathcal{S}$ (except for 4 source-only tasks that are in $\mathcal{S}$ only). This was done for various supervision budgets (columns) and maximum allowed order (rows) constraints. Still seeing transfers to some targets when the budget is 26 (full dictionary) means certain transfers became better than their fully supervised task-specific counterpart.

While Fig. 8 shows the structure and connectivity, Fig. 9 quantifies the results of taxonomy recommended transfer

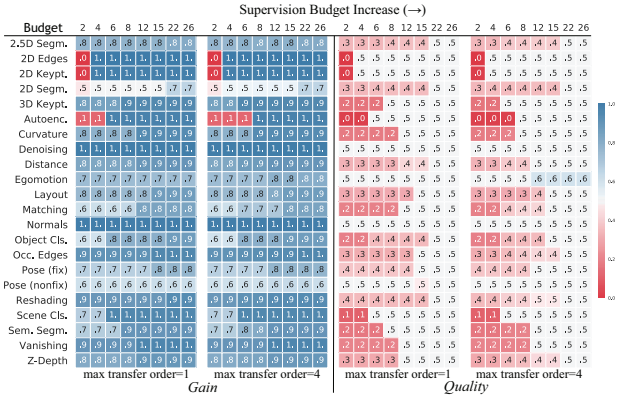


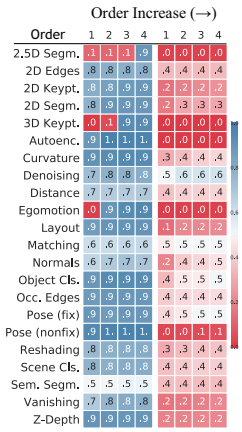
Figure 9: **Evaluation of taxonomy computed for solving the full task dictionary**. Gain (left) and Quality (right) values for each task using the policy suggested by the computed taxonomy, as the supervision budget increases(→). Shown for transfer orders 1 and 4.

policies by two metrics of *Gain* and *Quality*, defined as:

Gain: win rate (%) against a network trained from scratch using the same training data as transfer networks'. That is, the best that could be done if transfer learning was not utilized. This quantifies the *gained* value by transferring.

Quality: win rate (%) against a fully supervised network trained with 120k images (gold standard).

Red (0) and Blue (1) represent outperforming the reference method on none and all of test set images, respec-



Task	scratch	ImageNet[51]	Wang.[96]	Agrawal.[1]	Zamir.[100]	Zhang.[103]	Noroozi.[68]	full sup.	Taxonomy
Depth	88	88	93	89	88	84	86	43	-
	.03	.04	.04	.03	.04	.03	.03	**.02**	.02
Scene Cls.	80	52	83	74	74	71	75	15	-
	3.30	2.76	3.56	3.15	3.17	3.09	3.19	**2.23**	2.63
Sem. Segm.	78	79	82	85	76	78	84	21	-
	1.74	1.88	1.92	1.80	1.85	1.74	1.71	**1.42**	1.53
Object Cls.	79	54	82	76	75	76	76	34	-
	4.08	3.57	4.27	3.99	3.98	4.00	3.97	**3.26**	3.46
Normals	97	98	98	98	98	97	97	6	-
	.22	.30	.34	.28	.28	.23	.24	**.12**	.15
2.5D Segm.	80	93	92	89	90	84	87	40	-
	.21	.34	.34	.26	.29	.22	.24	**.16**	.17
Occ. Edges	93	96	95	93	94	93	94	42	-
	.16	.19	.18	.17	.18	.16	.17	**.12**	.13
Curvature	88	94	89	85	88	92	88	29	-
	.25	.28	.26	.25	.26	.26	.25	**.21**	.22
Egomotion	79	78	83	77	76	74	71	59	-
	8.60	8.58	9.26	8.41	8.34	8.15	7.94	7.32	**6.85**
Layout	80	76	85	79	77	78	70	36	-
	.66	.66	.85	.65	.65	.62	.54	**.37**	.41

Figure 10: **Generalization to Novel Tasks**. Each row shows a novel test task. Left: Gain and Quality values using the devised "all-for-one" transfer policies for novel tasks for orders 1-4. Right: Win rates (%) of the transfer policy over various self-supervised methods, ImageNet features, and scratch are shown in the colored rows. Note the large margin of win by taxonomy. The uncolored rows show corresponding loss values.

tively (so the transition Red→White→Blue is desirable. White (.5) represents equal performance to reference).

Each column in Fig. 9 shows a supervision budget. As apparent, good results can be achieved even when the supervision budget is notably smaller than the number of solved tasks, and as the budget increases, results improve (expected). Results are shown for 2 maximum allowed orders.

4.2. Generalization to Novel Tasks

The taxonomies in Sec. 4.1 were optimized for solving all tasks in the dictionary. In many situations, a practitioner is interested in a single task which even may not be in the dictionary. Here we evaluate how taxonomy transfers to a novel out-of-dictionary task with little data.

This is done in an all-for-one scenario where we put one task in $\mathcal{T}$ and all others in $\mathcal{S}$. The task in $\mathcal{T}$ is target-only and has no task-specific network. Its limited data (16k) is used to train small transfer networks to sources. This basically *localizes* where the target would be in the taxonomy.

Fig. 10 (left) shows the *Gain* and *Quality* of the transfer policy found by the BIP for each task. Fig. 10 (right) compares the taxonomy suggested policy against some of the best existing self-supervised methods [96, 103, 68, 100, 1], ImageNet FC7 features [51], training from scratch, and a fully supervised network (gold standard).

The results in Fig. 10 (right) are noteworthy. The large win margin for taxonomy shows that carefully selecting transfer policies depending on the target is superior to fixed transfers, such as the ones employed by self-supervised methods. ImageNet features which are the most popular off-the-shelf features in vision are also outperformed by those policies. Additionally, though the taxonomy transfer policies lose to fully supervised networks (gold standard) in

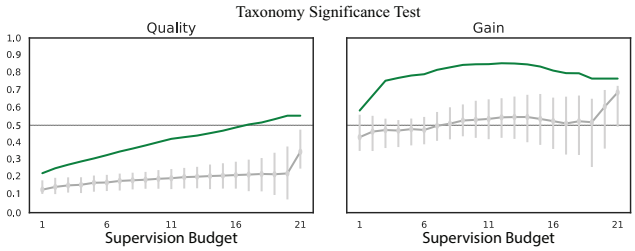


Figure 11: **Structure Significance.** Our taxonomy compared with random transfer policies (random feasible taxonomies that use the maximum allowable supervision budget). Y-axis shows *Quality* or *Gain*, and X-axis is the supervision budget. Green and gray represent our taxonomy and random connectivities, respectively. Error bars denote 5^{th}–95^{th} percentiles.

most cases, the results often get close with win rates in 40% range. These observations suggests the space has a rather predicable and strong structure. For graph visualization of the all-for-one taxonomy policies please see the supplementary material. The solver website allows generating the taxonomy for arbitrary sets of target-only tasks.

5. Significance Test of the Structure

The previous evaluations showed good transfer results in terms of *Quality* and *Gain*, but how crucial is it to use our taxonomy to choose smart transfers over just choosing *any* transfer? In other words, how *significant/strong* is the discovered structure of task space? Fig. 11 quantifies this by showing the performance of our taxonomy versus a large set of taxonomies with random connectivities. Our taxonomy outperformed all other connectivities by a large margin signifying both existence of a strong structure in the space as well as a good modeling of it by our approach. Complete experimental details is available in supplementary material.

5.1. Evaluation on MIT Places & ImageNet

To what extent are our findings dataset dependent, and would the taxonomy change if done on another dataset? We examined this by finding the ranking of all tasks for transferring to two target tasks of object classification and scene classification on our dataset. We then fine tuned our task-specific networks on other datasets (MIT Places [104] for scene classification, ImageNet [78] for object classification) and evaluated them on their respective test sets and metrics. Fig. 12 shows how the results correlate with taxonomy's ranking from our dataset. The Spearman's rho between the taxonomy ranking and the Top-1 ranking is 0.857 on Places and 0.823 on ImageNet showing a notable correlation. See supplementary material for complete experimental details.

5.2. Universality of the Structure

We employed a computational approach with various design choices. It is important to investigate how specific to those the discovered structure is. We did stability tests by

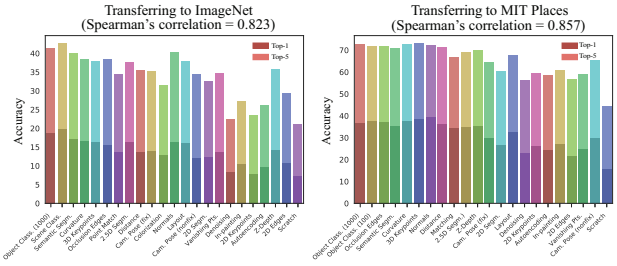


Figure 12: **Evaluating the discovered structure on other datasets: ImageNet [78] (left) for object classification and MIT Places [104] (right) for scene classification.** Y-axis shows accuracy on the external benchmark while bars on x-axis are ordered by taxonomy's predicted performance based on our dataset. A monotonically decreasing plot corresponds to preserving identical orders and perfect generalization.

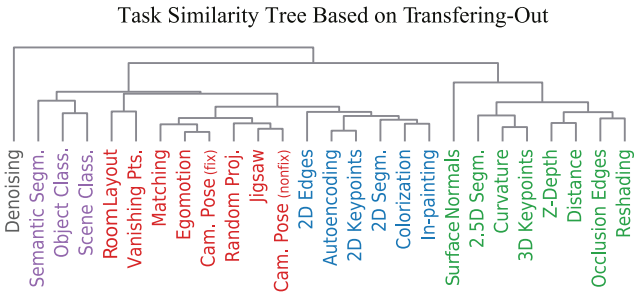


Figure 13: **Task Similarity Tree.** Agglomerative clustering of tasks based on their transferring-out patterns (i.e. using columns of normalized affinity matrix as task features). 3D, 2D, low dimensional geometric, and semantic tasks clustered together using a fully computational approach.

computing the variance in our output when making changes in one of the following system choices: **I. architecture of task-specific networks, II. architecture of transfer function networks, III. amount of data available for training transfer networks, IV. datasets, V. data splits, VI. choice of dictionary**. Overall, despite injecting large changes (e.g. varying the size of training data of transfer functions by 16x, size and architecture of task-specific networks and transfer networks by 4x), we found the outputs to be remarkably stable leading to almost no change in the output taxonomy computed on top. Detailed results and experimental setup of each tests are reported in the supplementary material.

5.3. Task Similarity Tree

Thus far we showed the task space has a structure, measured this structure, and presented its utility for transfer learning via devising transfer policies. This structure can be presented in other manners as well, e.g. via a metric of similarity across tasks. Figure 13 shows a similarity tree for the tasks in our dictionary. This is acquired from agglomerative clustering of the tasks based on their transferring-out behavior, i.e. using columns of normalized affinity matrix P as feature vectors for tasks. The tree shows how tasks would be hierarchically positioned w.r.t. to each other when measured based on providing information for solving other tasks; the closer two tasks, the more similar their role in transferring to other tasks. Notice that the 3D, 2D, low dimensional geometric, and semantic tasks are found to cluster together using a fully computational approach, which matches the intuitive expectations from the structure of task space. The transfer taxonomies devised by BIP are consistent with this tree as BIP picks the sources in a way that all of these modes are quantitatively best covered, subject to the given budget and desired target set.

6. Limitations and Discussion

We presented a method for modeling the space of visual tasks by way of transfer learning and showed its utility in reducing the need for supervision. The space of tasks is an interesting object of study in its own right and we have only scratched the surface in this regard. We also made a number of assumptions in the framework which should be noted.

Model Dependence: We used a computational approach and adopted neural networks as our function class. Though we validated the stability of the findings w.r.t various architectures and datasets, it should be noted that the findings are in principle model and data specific.

Compositionality: We performed the modeling via a set of common human-defined visual tasks. It is natural to consider a further compositional approach in which such common tasks are viewed as *observed samples* which are composed of computationally found latent subtasks.

Space Regularity: We performed modeling of a dense space via a sampled dictionary. Though we showed a good tolerance w.r.t. to the choice of dictionary and transferring to out-of-dictionary tasks, this outcome holds upon a proper sampling of the space as a function of its regularity. More formal studies on properties of the computed space is required for this to be provably guaranteed for a general case.

Transferring to Non-visual and Robotic Tasks: Given the structure of the space of visual tasks and demonstrated transferabilities to novel tasks, it is worthwhile to question how this can be employed to develop a perception module for solving downstream tasks which are not entirely visual, e.g. robotic manipulation, but entail solving a set of (a priori unknown) visual tasks.

Lifelong Learning: We performed the modeling in one go. In many cases, e.g. lifelong learning, the system is evolving and the number of mastered tasks constantly increase. Such scenarios require augmentation of the structure with expansion mechanisms based on new beliefs.

Acknowledgement: We acknowledge the support of NSF (DMS-1521608), MURI (1186514-1-TBCJE), ONR MURI (N00014-14-1-0671), Toyota(1191689-1-UDAWF), ONR MURI (N00014-13-1-0341), Nvidia, Tencent, a gift by Amazon Web Services, a Google Focused Research Award.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015. 8

[2] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016. 2

[3] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 4

[4] S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable bounds for learning some deep representations. In *International Conference on Machine Learning*, pages 584–592, 2014. 2

[5] Y. Aytar and A. Zisserman. Tabula rasa: Model transfer for object category detection. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2252–2259. IEEE, 2011. 2

[6] J. Baxter. A bayesian/information theoretic model of learning to learn viamultiple task sampling. *Mach. Learn.*, 28(1):7–39, July 1997. 3

[7] S. Ben-David and R. S. Borbely. A notion of task relatedness yielding provable multiple-task learning guarantees. *Machine Learning*, 73(3):273–287, Dec 2008. 3

[8] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 2

[9] P. Berkhin et al. A survey of clustering data mining techniques. *Grouping multidimensional data*, 25:71, 2006. 2

[10] E. Bienenstock, S. Geman, and D. Potter. Compositionality, mdl priors, and object recognition. In *Advances in neural information processing systems*, pages 838–844, 1997. 2

[11] H. Bilen and A. Vedaldi. Integrated perception with recurrent multi-task neural networks. In *Advances in neural information processing systems*, pages 235–243, 2016. 2

[12] J. Bingel and A. Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303*, 2017. 2

[13] O. Boiman and M. Irani. Similarity by composition. In *Advances in neural information processing systems*, pages 177–184, 2007. 2

[14] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 4

[15] Z. Chen and B. Liu. *Lifelong Machine Learning*. Morgan & Claypool Publishers, 2016. 2

[16] I. I. CPLEX. V12. 1: Users manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009. 5

[17] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. 2

[18] C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. *arXiv preprint arXiv:1708.07860*, 2017. 2

[19] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014. 2

[20] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016. 2

[21] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016. 2

[22] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010. 2

[23] A. Faktor and M. Irani. clustering by composition–unsupervised discovery of image categories. In *European Conference on Computer Vision*, pages 474–487. Springer, 2012. 2

[24] L. Fe-Fei et al. A bayesian approach to unsupervised one-shot learning of object categories. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1134–1141. IEEE, 2003. 2

[25] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006. 2

[26] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE international conference on computer vision*, pages 2960–2967, 2013. 2

[27] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017. 2

[28] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. *CoRR*, abs/1603.00448, 2016. 2

[29] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 512–519. IEEE, 2016. 2

[30] C. Finn, T. Yu, J. Fu, P. Abbeel, and S. Levine. Generalizing skills with semi-supervised reinforcement learning. *CoRR*, abs/1612.00429, 2016. 2

[31] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. *CoRR*, abs/1709.04905, 2017. 2

[32] I. K. Fodor. A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Lab., CA (US), 2002. 2

[33] R. M. French. Catastrophic forgetting in connectionist networks: Causes, consequences and solutions. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. 2

[34] R. Ge. *Provable algorithms for machine learning problems*. PhD thesis, Princeton University, 2013. 2

[35] S. Geman, D. F. Potter, and Z. Chi. Composition systems. *Quarterly of Applied Mathematics*, 60(4):707–736, 2002. 2

[36] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 999–1006. IEEE, 2011. 2

[37] A. Gopnik, C. Glymour, D. Sobel, L. Schulz, T. Kushnir, and D. Danks. A theory of causal learning in children: Causal maps and bayes nets. 111:3–32, 02 2004. 2

[38] A. Gopnik, C. Glymour, D. M. Sobel, L. E. Schulz, T. Kushnir, and D. Danks. A theory of causal learning in children: causal maps and bayes nets. *Psychological review*, 111(1):3, 2004. 2

[39] A. Gopnik, A. N. Meltzoff, and P. K. Kuhl. *The scientist in the crib: Minds, brains, and how children learn.* William Morrow & Co, 1999. 2

[40] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. 2

[41] I. Gurobi Optimization. Gurobi optimizer reference manual, 2016. 5, 6

[42] K. Henry. The theory and applications of homomorphic cryptography. Master's thesis, University of Waterloo, 2008. 2

[43] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 4

[44] J. Hoffman, T. Darrell, and K. Saenko. Continuous manifold based adaptation for evolving visual domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 867–874, 2014. 2

[45] Y. Hoshen and S. Peleg. Visual learning of arithmetic operations. *CoRR*, abs/1506.02264, 2015. 2

[46] F. Hu, G.-S. Xia, J. Hu, and L. Zhang. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sensing*, 7(11):14680–14707, 2015. 2

[47] I.-H. Jhuo, D. Liu, D. Lee, and S.-F. Chang. Robust visual domain adaptation with low-rank reconstruction. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2168–2175. IEEE, 2012. 2

[48] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 2

[49] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[50] I. Kokkinos. Ubernet: Training auniversal'convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *arXiv preprint arXiv:1609.02132*, 2016. 2

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 8

[52] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1785–1792. IEEE, 2011. 2

[53] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016. 7

[54] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 2

[55] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, pages 1–101, 2016. 2

[56] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. *arXiv preprint arXiv:1611.07709*, 2016. 4

[57] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 4

[58] F. Liu, G. Lin, and C. Shen. CRF learning with CNN features for image segmentation. *CoRR*, abs/1503.08263, 2015. 2

[59] Z. Luo, Y. Zou, J. Hoffman, and L. F. Fei-Fei. Label efficient learning of transferable representations acrosss domains and tasks. In *Advances in Neural Information Processing Systems*, pages 164–176, 2017. 2

[60] M. M. H. Mahmud. *On Universal Transfer Learning*, pages 135–149. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. 3

[61] J. Malik, P. Arbeláez, J. Carreira, K. Fragkiadaki, R. Girshick, G. Gkioxari, S. Gupta, B. Hariharan, A. Kar, and S. Tulsiani. The three rs of computer vision: Recognition, reconstruction and reorganization. *Pattern Recognition Letters*, 72:4–14, 2016. 2

[62] N. Masuda, M. A. Porter, and R. Lambiotte. Random walks and diffusion on networks. *Physics Reports*, 716-717:1 – 58, 2017. Random walks and diffusion on networks. 5

[63] M. Mccloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24, 1989. 2

[64] L. Mihalkova, T. Huynh, and R. J. Mooney. Mapping and revising markov logic networks for transfer learning. In *AAAI*, volume 7, pages 608–614, 2007. 2

[65] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013. 2

[66] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016. 2

[67] A. Niculescu-Mizil and R. Caruana. Inductive transfer for bayesian network structure learning. In *Artificial Intelligence and Statistics*, pages 339–346, 2007. 2

[68] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 2, 8

[69] M. Noroozi, H. Pirsiavash, and P. Favaro. Representation learning by learning to count. *arXiv preprint arXiv:1708.06734*, 2017. 2

[70] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013. 2

[71] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 31(4):30, 2012. 2

[72] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016. 2

[73] A. Pentina and C. H. Lampert. Multi-task learning with labeled and unlabeled tasks. *stat*, 1050:1, 2017. 2

[74] J. Piaget and M. Cook. *The origins of intelligence in children*, volume 8. International Universities Press New York, 1952. 2

[75] L. Y. Pratt. Discriminability-based transfer between neural networks. In *Advances in neural information processing systems*, pages 204–211, 1993. 2

[76] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *International Conference on Computer Vision (ICCV)*, 2017. 2

[77] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000. 2

[78] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 2, 4, 8, 9

[79] R. W. Saaty. The analytic hierarchy process – what it is and how it is used. *Mathematical Modeling*, 9(3-5):161–176, 1987. 5

[80] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. *Computer Vision–ECCV 2010*, pages 213–226, 2010. 2

[81] R. Salakhutdinov, J. Tenenbaum, and A. Torralba. One-shot learning with a hierarchical nonparametric bayesian model. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 195–206, 2012. 2

[82] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. 2

[83] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014. 2

[84] D. L. Silver and K. P. Bennett. Guest editors introduction: special issue on inductive transfer learning. *Machine Learning*, 73(3):215–220, 2008. 2

[85] D. L. Silver, Q. Yang, and L. Li. Lifelong machine learning systems: Beyond learning algorithms. In *in AAAI Spring Symposium Series*, 2013. 2

[86] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, pages 935–943, 2013. 2

[87] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 2

[88] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. 2

[89] J. B. Tenenbaum and T. L. Griffiths. Generalization, similarity, and bayesian inference. *Behavioral and Brain Sciences*, 24(4):629640, 2001. 2

[90] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011. 2

[91] J. B. Tenenbaum, C. Kemp, and P. Shafto. Theory-based bayesian models of inductive learning and reasoning. In *Trends in Cognitive Sciences*, pages 309–318, 2006. 2

[92] D. G. R. Tervo, J. B. Tenenbaum, and S. J. Gershman. Toward the neural implementation of structure learning. *Current opinion in neurobiology*, 37:99–105, 2016. 2

[93] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*, pages 1553–1561, 2017. 2

[94] S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 2012. 2

[95] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950. 2

[96] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015. 8

[97] X. Wang, K. He, and A. Gupta. Transitive invariance for self-supervised visual representation learning. *arXiv preprint arXiv:1708.02901*, 2017. 2

[98] T. Winograd. *Thinking machines: Can there be? Are we*, volume 200. University of California Press, Berkeley, 1991. 2

[99] J. Yang, R. Yan, and A. G. Hauptmann. Adapting svm classifiers to data with shifted distributions. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 69–76. IEEE, 2007. 2

[100] A. R. Zamir, T. Wekel, P. Agrawal, C. Wei, J. Malik, and S. Savarese. Generic 3d representation via pose estimation and matching. In *European Conference on Computer Vision*, pages 535–553. Springer, 2016. 2, 8

[101] A. R. Zamir, F. Xia, J. He, A. Sax, J. Malik, and S. Savarese. Gibson Env: Real-world perception for embodied agents. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018. 4

[102] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. 2

[103] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016. 2, 8

[104] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014. 2, 4, 8, 9

GANimation: Anatomically-aware Facial Animation from a Single Image

Albert Pumarola[1], Antonio Agudo[1], Aleix M. Martinez[2],
Alberto Sanfeliu[1], Francesc Moreno-Noguer[1]

[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC, 08028, Barcelona, Spain
[2]The Ohio State University, Columbus, OH 43210, USA

Abstract. Recent advances in Generative Adversarial Networks (GANs) have shown impressive results for task of facial expression synthesis. The most successful architecture is StarGAN [4], that conditions GANs' generation process with images of a specific domain, namely a set of images of persons sharing the same expression. While effective, this approach can only generate a discrete number of expressions, determined by the content of the dataset. To address this limitation, in this paper, we introduce a novel GAN conditioning scheme based on Action Units (AU) annotations, which describes in a continuous manifold the anatomical facial movements defining a human expression. Our approach allows controlling the magnitude of activation of each AU and combine several of them. Additionally, we propose a fully unsupervised strategy to train the model, that only requires images annotated with their activated AUs, and exploit attention mechanisms that make our network robust to changing backgrounds and lighting conditions. Extensive evaluation show that our approach goes beyond competing conditional generators both in the capability to synthesize a much wider range of expressions ruled by anatomically feasible muscle movements, as in the capacity of dealing with images in the wild.

Keywords: GANs, Face Animation, Action-Unit Condition.

1 Introduction

Being able to automatically animate the facial expression from a single image would open the door to many new exciting applications in different areas, including the movie industry, photography technologies, fashion and e-commerce business, to name but a few. As Generative and Adversarial Networks have become more prevalent, this task has experienced significant advances, with architectures such as StarGAN [4], which is able not only to synthesize novel expressions, but also to change other attributes of the face, such as age, hair color or gender. Despite its generality, StarGAN can only change a particular aspect of a face among a discrete number of attributes defined by the annotation granularity of the dataset. For instance, for the facial expression synthesis task,

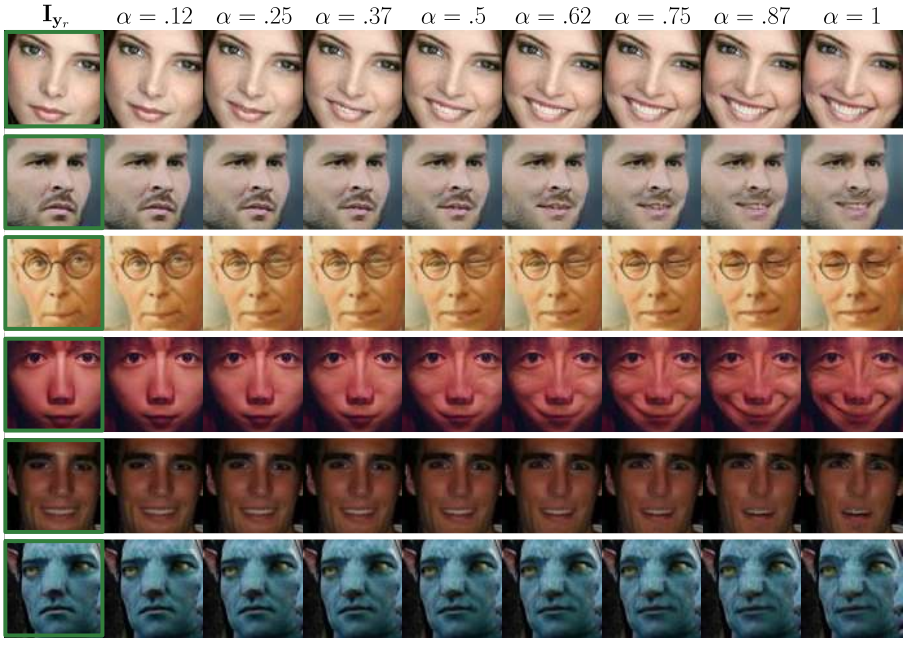


Fig. 1. Facial animation from a single image. We propose an anatomically coherent approach that is not constrained to a discrete number of expressions and can animate a given image and render novel expressions in a continuum. In these examples, we are given solely the left-most input image $\mathbf{I}_{\mathbf{y}_r}$ (highlighted by a green square), and the parameter α controls the degree of activation of the target action units involved in a smiling-like expression. Additionally, our system can handle images with unnatural illumination conditions, such as the example in the bottom row.

[4] is trained on the RaFD [16] dataset which has only 8 binary labels for facial expressions, namely sad, neutral, angry, contemptuous, disgusted, surprised, fearful and happy.

Facial expressions, however, are the result of the combined and coordinated action of facial muscles that cannot be categorized in a discrete and low number of classes. Ekman and Friesen [6] developed the Facial Action Coding System (FACS) for describing facial expressions in terms of the so-called Action Units (AUs), which are anatomically related to the contractions of specific facial muscles. Although the number of action units is relatively small (30 AUs were found to be anatomically related to the contraction of specific facial muscles), more than 7,000 different AU combinations have been observed [30]. For example, the facial expression for *fear* is generally produced with activations: Inner Brow Raiser (AU1), Outer Brow Raiser (AU2), Brow Lowerer (AU4), Upper Lid Raiser (AU5), Lid Tightener (AU7), Lip Stretcher (AU20) and Jaw Drop (AU26) [5]. Depending on the magnitude of each AU, the expression will transmit the emotion of fear to a greater or lesser extent.

In this paper we aim at building a model for synthetic facial animation with the level of expressiveness of FACS, and being able to generate anatomically-

aware expressions in a continuous domain, without the need of obtaining any facial landmarks [36]. For this purpose we leverage on the recent EmotioNet dataset [3], which consists of one million images of facial expressions (we use 200,000 of them) of emotion in the wild annotated with discrete AUs activations [1]. We build a GAN architecture which, instead of being conditioned with images of a specific domain as in [4], it is conditioned on a one-dimensional vector indicating the presence/absence and the magnitude of each action unit. We train this architecture in an unsupervised manner that only requires images with their activated AUs. To circumvent the need for pairs of training images of the same person under different expressions, we split the problem in two main stages. First, we consider an AU-conditioned bidirectional adversarial architecture which, given a single training photo, initially renders a new image under the desired expression. This synthesized image is then rendered-back to the original pose, hence being directly comparable to the input image. We incorporate very recent losses to assess the photorealism of the generated image. Additionally, our system also goes beyond state-of-the-art in that it can handle images under changing backgrounds and illumination conditions. We achieve this by means of an attention layer that focuses the action of the network only in those regions of the image that are relevant to convey the novel expression.

As a result, we build an anatomically coherent facial expression synthesis method, able to render images in a continuous domain, and which can handle images in the wild with complex backgrounds and illumination conditions. As we will show in the results section, it compares favorably to other conditioned-GANs schemes, both in terms of the visual quality of the results, and the possibilities of generation. Figure 1 shows some example of the results we obtain, in which given one input image, we gradually change the magnitude of activation of the AUs used to produce a smile.

2 Related Work

Generative Adversarial Networks. GANs are a powerful class of generative models based on game theory. A typical GAN optimization consists in simultaneously training a generator network to produce realistic fake samples and a discriminator network trained to distinguish between real and fake data. This idea is embedded by the so-called *adversarial loss*. Recent works [1,9] have shown improved stability relaying on the continuous Earth Mover Distance metric, which we shall use in this paper to train our model. GANs have been shown to produce very realistic images with a high level of detail and have been successfully used for image translation [38,10,13], face generation [12,28] , super-resolution imaging [34,18], indoor scene modeling [12,33] and human poses editing [27].

Conditional GANs. An active area of research is designing GAN models that incorporate conditions and constraints into the generation process. Prior studies have explored combining several conditions, such as text descriptions [29,39,37]

[1] The dataset was re-annotated with [2] to obtain continuous activation annotations.

and class information [24,23]. Particularly interesting for this work are those methods exploring image based conditioning as in image super-resolution [18], future frame prediction [22], image in-painting [25], image-to-image translation [10] and multi-target domain transfer [4].

Unpaired Image-to-Image Translation. As in our framework, several works have also tackled the problem of using unpaired training data. First attempts [21] relied on Markov random field priors for Bayesian based generation models using images from the marginal distributions in individual domains. Others explored enhancing GANS with Variational Auto-Encoder strategies [21,15]. Later, several works [25,19] have exploited the idea of driving the system to produce mappings transforming the style without altering the original input image content. Our approach is more related to those works exploiting cycle consistency to preserve key attributes between the input and the mapped image, such as CycleGAN [38], DiscoGAN [13] and StarGAN [4].

Face Image Manipulation. Face generation and editing is a well-studied topic in computer vision and generative models. Most works have tackled the task on attribute editing [17,26,31] trying to modify attribute categories such as adding glasses, changing color hair, gender swapping and aging. The works that are most related to ours are those synthesizing facial expressions. Early approaches addressed the problem using mass-and-spring models to physically approximate skin and muscle movement [7]. The problem with this approach is that is difficult to generate natural looking facial expressions as there are many subtle skin movements that are difficult to render with simple spring models. Another line of research relied on 2D and 3D morphings [35], but produced strong artifacts around the region boundaries and was not able to model illumination changes.

More recent works [4,24,20] train highly complex convolutional networks able to work with images in the wild. However, these approaches have been conditioned on discrete emotion categories (e.g., happy, neutral, and sad). Instead, our model resumes the idea of modeling skin and muscles, but we integrate it in modern deep learning machinery. More specifically, we learn a GAN model conditioned on a continuous embedding of muscle movements, allowing to generate a large range of anatomically possible face expressions as well as smooth facial movement transitions in video sequences.

3 Problem Formulation

Let us define an input RGB image as $\mathbf{I}_{\mathbf{y}_r} \in \mathbb{R}^{H \times W \times 3}$, captured under an arbitrary facial expression. Every gesture expression is encoded by means of a set of N action units $\mathbf{y}_r = (y_1, \ldots, y_N)^\top$, where each y_n denotes a normalized value between 0 and 1 to module the magnitude of the n-th action unit. It is worth pointing out that thanks to this continuous representation, a natural interpolation can be done between different expressions, allowing to render a wide range of realistic and smooth facial expressions.

Our aim is to learn a mapping $\mathcal{M}$ to translate $\mathbf{I}_{\mathbf{y}_r}$ into an output image $\mathbf{I}_{\mathbf{y}_g}$ conditioned on an action-unit target $\mathbf{y}_g$, i.e., we seek to estimate the mapping

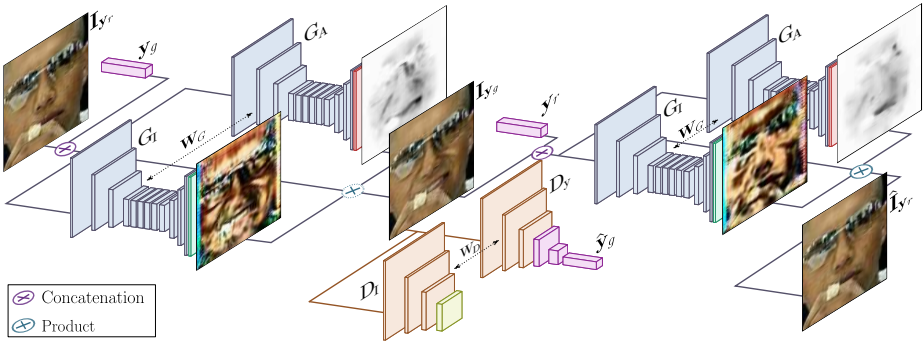


Fig. 2. Overview of our approach to generate photo-realistic conditioned images. The proposed architecture consists of two main blocks: a generator G to regress attention and color masks; and a critic D to evaluate the generated image in its photorealism D_I and expression conditioning fulfillment $\hat{\mathbf{y}}_g$. Note that our systems does not require supervision, i.e., no pairs of images of the same person with different expressions, nor the target image $\mathbf{I}_{\mathbf{y}_g}$ are assumed to be known.

$\mathcal{M} : (\mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_g) \rightarrow \mathbf{I}_{\mathbf{y}_g}$. To this end, we propose to train $\mathcal{M}$ in an unsupervised manner, using M training triplets $\{\mathbf{I}^m_{\mathbf{y}_r}, \mathbf{y}^m_r, \mathbf{y}^m_g\}_{m=1}^M$, where the target vectors $\mathbf{y}^m_g$ are randomly generated. Importantly, we neither require pairs of images of the same person under different expressions, nor the expected target image $\mathbf{I}_{\mathbf{y}_g}$.

4 Our Approach

This section describes our novel approach to generate photo-realistic conditioned images, which, as shown in Fig. 2, consists of two main modules. On the one hand, a generator $G(\mathbf{I}_{\mathbf{y}_r}|\mathbf{y}_g)$ is trained to realistically transform the facial expression in image $\mathbf{I}_{\mathbf{y}_r}$ to the desired $\mathbf{y}_g$. Note that G is applied twice, first to map the input image $\mathbf{I}_{\mathbf{y}_r} \rightarrow \mathbf{I}_{\mathbf{y}_g}$, and then to render it back $\mathbf{I}_{\mathbf{y}_g} \rightarrow \hat{\mathbf{I}}_{\mathbf{y}_r}$. On the other hand, we use a WGAN-GP [9] based critic $D(\mathbf{I}_{\mathbf{y}_g})$ to evaluate the quality of the generated image as well as its expression.

4.1 Network Architecture

Generator. Let G be the generator block. Since it will be applied bidirectionally (i.e., to map either input image to desired expression and vice-versa) in the following discussion we use subscripts o and f to indicate *origin* and *final*.

Given the image $\mathbf{I}_{\mathbf{y}_o} \in \mathbb{R}^{H \times W \times 3}$ and the N-vector $\mathbf{y}_f$ encoding the desired expression, we form the input of generator as a concatenation $(\mathbf{I}_{\mathbf{y}_o}, \mathbf{y}_o) \in \mathbb{R}^{H \times W \times (N+3)}$, where $\mathbf{y}_o$ has been represented as N arrays of size $H \times W$.

One key ingredient of our system is to make G focus only on those regions of the image that are responsible of synthesizing the novel expression and keep the rest elements of the image such as hair, glasses, hats or jewelery untouched. For this purpose, we have embedded an attention mechanism to the generator.

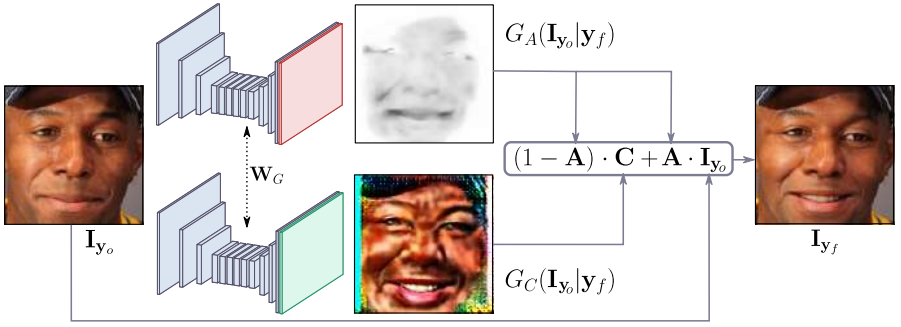


Fig. 3. Attention-based generator. Given an input image and the target expression, the generator regresses and attention mask $\mathbf{A}$ and an RGB color transformation $\mathbf{C}$ over the entire image. The attention mask defines a per pixel intensity specifying to which extend each pixel of the original image will contribute in the final rendered image.

Concretely, instead of regressing a full image, our generator outputs two masks, a color mask $\mathbf{C}$ and attention mask $\mathbf{A}$. The final image can be obtained as:

$$\mathbf{I}_{\mathbf{y}_f} = (1-\mathbf{A})\cdot\mathbf{C} + \mathbf{A}\cdot\mathbf{I}_{\mathbf{y}_o}\ , \tag{1}$$

where $\mathbf{A} = G_A(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f) \in \{0,\dots,1\}^{H\times W}$ and $\mathbf{C} = G_C(\mathbf{I}_{y_o}|\mathbf{y}_f) \in \mathbb{R}^{H\times W\times 3}$. The mask $\mathbf{A}$ indicates to which extend each pixel of the $\mathbf{C}$ contributes to the output image $\mathbf{I}_{\mathbf{y}_f}$. In this way, the generator does not need to render static elements, and can focus exclusively on the pixels defining the facial movements, leading to sharper and more realistic synthetic images. This process is depicted in Fig. 3.

Conditional Critic. This is a network trained to evaluate the generated images in terms of their photo-realism and desired expression fulfillment. The structure of $D(\mathbf{I})$ resembles that of the PatchGan [10] network mapping from the input image $\mathbf{I}$ to a matrix $\mathbf{Y}_\mathrm{I} \in \mathbb{R}^{H/2^6\times W/2^6}$, where $\mathbf{Y}_\mathrm{I}[i,j]$ represents the probability of the overlapping patch ij to be real. Also, to evaluate its conditioning, on top of it we add an auxiliary regression head that estimates the AUs activations $\hat{\mathbf{y}} = (\hat{y}_1,\dots,\hat{y}_N)^\top$ in the image.

4.2 Learning the Model

The loss function we define contains four terms, namely an *image adversarial loss* [1] with the modification proposed by Gulrajani *et al.* [9] that pushes the distribution of the generated images to the distribution of the training images; the *attention loss* to drive the attention masks to be smooth and prevent them from saturating; the *conditional expression loss* that conditions the expression of the generated images to be similar to the desired one; and the *identity loss* that favors to preserve the person texture identity.

Image Adversarial Loss. In order to learn the parameters of the generator G, we use the modification of the standard GAN algorithm [8] proposed by WGAN-GP [9]. Specifically, the original GAN formulation is based on the

Jensen-Shannon (JS) divergence loss function and aims to maximize the probability of correctly classifying real and rendered images while the generator tries to foul the discriminator. This loss is potentially not continuous with respect to the generators parameters and can locally saturate leading to vanishing gradients in the discriminator. This is addressed in WGAN [1] by replacing JS with the continuous Earth Mover Distance. To maintain a Lipschitz constraint, WGAN-GP [9] proposes to add a gradient penalty for the critic network computed as the norm of the gradients with respect to the critic input.

Formally, let $\mathbf{I}_{\mathbf{y}_o}$ be the input image with the initial condition $\mathbf{y}_o$, $\mathbf{y}_f$ the desired final condition, $\mathbb{P}_o$ the data distribution of the input image, and $\mathbb{P}_{\widetilde{I}}$ the random interpolation distribution. Then, the *critic loss* $\mathcal{L}_{\mathrm{I}}(G, D_{\mathrm{I}}, \mathbf{I}_{\mathbf{y}_o}, \mathbf{y}_f)$ we use is:

$$\mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}[D_{\mathrm{I}}(G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f))] - \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[D_{\mathrm{I}}(\mathbf{I}_{\mathbf{y}_o})\right] + \lambda_{\mathrm{gp}}\mathbb{E}_{\widetilde{I}\sim\mathbb{P}_{\widetilde{I}}}\left[(\|\nabla_{\widetilde{I}} D_{\mathrm{I}}(\widetilde{I})\|_2 - 1)^2\right],$$

where λ_{gp} is a penalty coefficient.

Attention Loss. When training the model we do not have ground-truth annotation for the attention masks $\mathbf{A}$. Similarly as for the color masks $\mathbf{C}$, they are learned from the resulting gradients of the critic module and the rest of the losses. However, the attention masks can easily saturate to 1 which makes that $\mathbf{I}_{\mathbf{y}_o} = G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f)$, that is, the generator has no effect. To prevent this situation, we regularize the mask with a l_2-weight penalty. Also, to enforce smooth spatial color transformation when combining the pixel from the input image and the color transformation $\mathbf{C}$, we perform a *Total Variation Regularization* over $\mathbf{A}$. The attention loss $\mathcal{L}_{\mathrm{A}}(G, \mathbf{I}_{\mathbf{y}_o}, \mathbf{y}_f)$ can therefore be defined as:

$$\lambda_{\mathrm{TV}}\mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\sum_{i,j}^{H,W}\left[(\mathbf{A}_{i+1,j} - \mathbf{A}_{i,j})^2 + (\mathbf{A}_{i,j+1} - \mathbf{A}_{i,j})^2\right]\right] + \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\|\mathbf{A}\|_2\right] \quad (2)$$

where $\mathbf{A} = G_A(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f)$ and $\mathbf{A}_{i,j}$ is the i, j entry of $\mathbf{A}$. λ_{TV} is a penalty coefficient.

Conditional Expression Loss. While reducing the *image adversarial loss*, the generator must also reduce the error produced by the AUs regression head on top of D. In this way, G not only learns to render realistic samples but also learns to satisfy the target facial expression encoded by $\mathbf{y}_f$. This loss is defined with two components: an AUs regression loss with fake images used to optimize G, and an AUs regression loss of real images used to learn the regression head on top of D. This loss $\mathcal{L}_{\mathrm{y}}(G, D_{\mathrm{y}}, \mathbf{I}_{\mathbf{y}_o}, \mathbf{y}_o, \mathbf{y}_f)$ is computed as:

$$\mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\|D_{\mathrm{y}}(G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f))] - \mathbf{y}_f\|_2^2\right] + \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\|D_{\mathrm{y}}(\mathbf{I}_{\mathbf{y}_o}) - \mathbf{y}_o\|_2^2\right]. \quad (3)$$

Identity Loss. With the previously defined losses the generator is enforced to generate photo-realistic face transformations. However, without ground-truth supervision, there is no constraint to guarantee that the face in both the input and output images correspond to the same person. Using a *cycle consistency*

loss [38] we force the generator to maintain the identity of each individual by penalizing the difference between the original image $\mathbf{I}_{\mathbf{y}_o}$ and its reconstruction:

$$\mathcal{L}_{\text{idt}}(G, \mathbf{I}_{\mathbf{y}_o}, \mathbf{y}_o, \mathbf{y}_f) = \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o} \sim \mathbb{P}_o} \left[\| G(G(\mathbf{I}_{\mathbf{y}_o} | \mathbf{y}_f) | \mathbf{y}_o) - \mathbf{I}_{\mathbf{y}_o} \|_1 \right]. \tag{4}$$

To produce realistic images it is critical for the generator to model both low and high frequencies. Our *PatchGan* based critic D_{I} already enforces high-frequency correctness by restricting our attention to the structure in local image patches. To also capture low-frequencies it is sufficient to use l_1-norm. In preliminary experiments, we also tried replacing l_1-norm with a more sophisticated *Perceptual* [11] loss, although we did not observe improved performance.

Full Loss. To generate the target image $\mathbf{I}_{\mathbf{y}_g}$, we build a loss function $\mathcal{L}$ by linearly combining all previous partial losses:

$$\begin{aligned} \mathcal{L} =& \mathcal{L}_{\text{I}}(G, D_{\text{I}}, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_g) + \lambda_{\text{y}} \mathcal{L}_{\text{y}}(G, D_{\text{y}}, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_r, \mathbf{y}_g) \\ &+ \lambda_{\text{A}} \left(\mathcal{L}_{\text{A}}(G, \mathbf{I}_{\mathbf{y}_g}, \mathbf{y}_r) + \mathcal{L}_{\text{A}}(G, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_g) \right) + \lambda_{\text{idt}} \mathcal{L}_{\text{idt}}(G, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_r, \mathbf{y}_g), \end{aligned} \tag{5}$$

where λ_{A}, λ_{y} and λ_{idt} are the hyper-parameters that control the relative importance of every loss term. Finally, we can define the following minimax problem:

$$G^\star = \arg \min_G \max_{D \in \mathcal{D}} \mathcal{L} \ , \tag{6}$$

where $G^\star$ draws samples from the data distribution. Additionally, we constrain our discriminator D to lie in $\mathcal{D}$, that represents the set of 1-Lipschitz functions.

5 Implementation Details

Our generator builds upon the variation of the network from Johnson *et al.* [11] proposed by [38] as it proved to achieve impressive results for image-to-image mapping. We have slightly modified it by substituting the last convolutional layer with two parallel convolutional layers, one to regress the color mask $\mathbf{C}$ and the other to define the attention mask $\mathbf{A}$. We also observed that changing batch normalization in the generator by instance normalization improved training stability. For the critic we have adopted the *PatchGan* architecture of [10], but removing feature normalization. Otherwise, when computing the gradient penalty, the norm of the critic's gradient would be computed with respect to the entire batch and not with respect to each input independently.

The model is trained on the EmotioNet dataset [3]. We use a subset of 200,000 samples (over 1 million) to reduce training time. We use Adam [14] with learning rate of 0.0001, beta1 0.5, beta2 0.999 and batch size 25. We train for 30 epochs and linearly decay the rate to zero over the last 10 epochs. Every 5 optimization steps of the critic network we perform a single optimization step of the generator. The weight coefficients for the loss terms in Eq. (5) are set to $\lambda_{\text{gp}} = 10$, $\lambda_{\text{A}} = 0.1$, $\lambda_{\text{TV}} = 0.0001$, $\lambda_{\text{y}} = 4000$, $\lambda_{\text{idt}} = 10$. To improve stability we tried updating the critic using a buffer with generated images in different updates of the generator as proposed in [32] but we did not observe performance improvement. The model takes two days to train with a single GeForce® GTX 1080 Ti GPU.

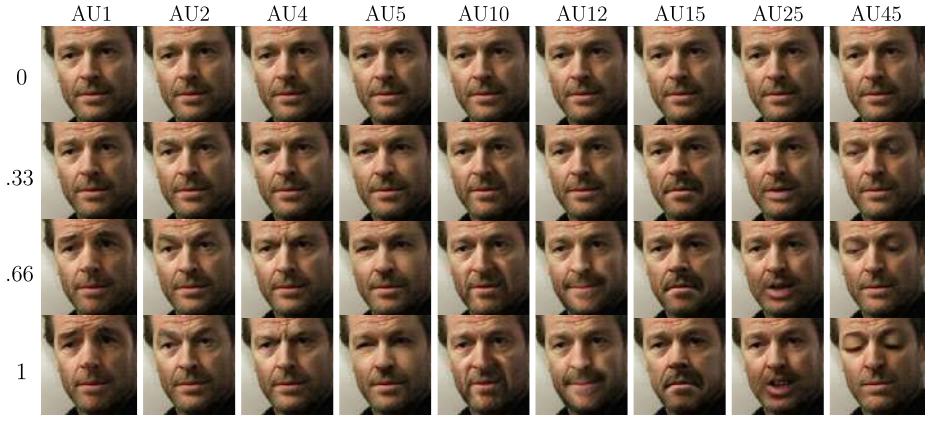


Fig. 4. Single AUs edition. Specific AUs are activated at increasing levels of intensity (from 0.33 to 1). The first row corresponds to a zero intensity application of the AU which correctly produces the original image in all cases.

6 Experimental Evaluation

This section provides a thorough evaluation of our system. We first test the main component, namely the single and multiple AUs editing. We then compare our model against current competing techniques in the task of discrete emotions editing and demonstrate our model's ability to deal with images in the wild and its capability to generate a wide range of anatomically coherent face transformations. Finally, we discuss the model's limitations and failure cases.

It is worth noting that in some of the experiments the input faces are not cropped. In this cases we first use a detector [2] to localize and crop the face, apply the expression transformation to that area with Eq. (1), and finally place the generated face back to its original position in the image. The attention mechanism guaranties a smooth transition between the morphed cropped face and the original image. As we shall see later, this three steps process results on higher resolution images compared to previous models. Supplementary material can be found on `http://www.albertpumarola.com/research/GANimation/`.

6.1 Single Action Units Edition

We first evaluate our model's ability to activate AUs at different intensities while preserving the person's identity. Figure 4 shows a subset of 9 AUs individually transformed with four levels of intensity (0, 0.33, 0.66, 1). For the case of 0 intensity it is desired not to change the corresponding AU. The model properly handles this situation and generates an identical copy of the input image for every case. The ability to apply an identity transformation is essential to ensure that non-desired facial movement will not be introduced.

[2] We use the face detector from `https://github.com/ageitgey/face_recognition`.

Fig. 5. Attention Model. Details of the intermediate attention mask **A** (first row) and the color mask **C** (second row). The bottom row images are the synthesized expressions. Darker regions of the attention mask **A** show those areas of the image more relevant for each specific AU. Brighter areas are retained from the original image.

For the non-zero cases, it can be observed how each AU is progressively accentuated. Note the difference between generated images at intensity 0 and 1. The model convincingly renders complex facial movements which in most cases are difficult to distinguish from real images. It is also worth mentioning that the independence of facial muscle cluster is properly learned by the generator. AUs relative to the eyes and half-upper part of the face (AUs 1, 2, 4, 5, 45) do not affect the muscles of the mouth. Equivalently, mouth related transformations (AUs 10, 12, 15, 25) do not affect eyes nor eyebrow muscles.

Fig. 5 displays, for the same experiment, the attention **A** and color **C** masks that produced the final result $\mathbf{I}_{\mathbf{y}_g}$. Note how the model has learned to focus its attention (darker area) onto the corresponding AU in an unsupervised manner. In this way, it relieves the color mask from having to accurately regress each pixel value. Only the pixels relevant to the expression change are carefully estimated, the rest are just noise. For example, the attention is clearly obviating background pixels allowing to directly copy them from the original image. This is a key ingredient to later being able to handle images in the wild (see Section 6.5).

6.2 Simultaneous Edition of Multiple AUs

We next push the limits of our model and evaluate it in editing multiple AUs. Additionally, we also assess its ability to interpolate between two expressions. The results of this experiment are shown in Fig. 1, the first column is the original image with expression $\mathbf{y}_r$, and the right-most column is a synthetically generated image conditioned on a target expression $\mathbf{y}_g$. The rest of columns result from evaluating the generator conditioned with a linear interpolation of the original and target expressions: $\alpha\mathbf{y}_g + (1-\alpha)\mathbf{y}_r$. The outcomes show a very remarkable smooth an consistent transformation across frames. We have intentionally selected challenging samples to show robustness to light conditions and even, as in the case of the avatar, to non-real world data distributions which were not previously seen by the model. These results are encouraging to further extend the model to video generation in future works.

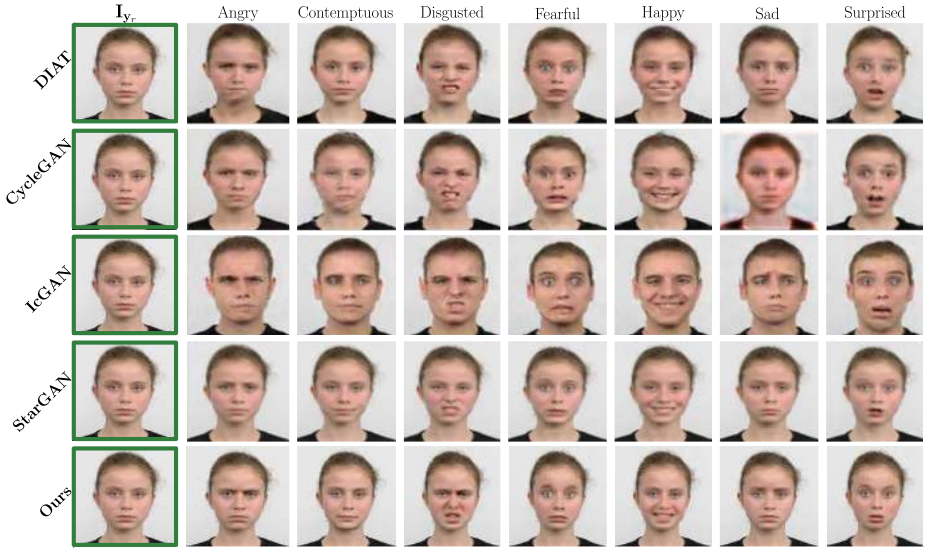


Fig. 6. Qualitative comparison with state-of-the-art. Facial Expression Synthesis results for: DIAT [20], CycleGAN [28], IcGAN [26] and StarGAN [4]; and ours. In all cases, we represent the input image and seven different facial expressions. As it can be seen, our solution produces the best trade-off between visual accuracy and spatial resolution. Some of the results of StarGAN, the best current approach, show certain level of blur. Images of previous models were taken from [4].

6.3 Discrete Emotions Editing

We next compare our approach, against the baselines DIAT [20], CycleGAN [28], IcGAN [26] and StarGAN [4]. For a fair comparison, we adopt the results of these methods trained by the most recent work, StarGAN, on the task of rendering discrete emotions categories (e.g., happy, sad and fearful) in the RaFD dataset [16]. Since DIAT [20] and CycleGAN [28] do not allow conditioning, they were independently trained for every possible pair of source/target emotions. We next briefly discuss the main aspects of each approach:

DIAT [20]. Given an input image $x \in X$ and a reference image $y \in Y$, DIAT learns a GAN model to render the attributes of domain Y in the image x while conserving the person's identity. It is trained with the classic *adversarial loss* and a *cycle loss* $\|x - G_{Y \to X}(G_{X \to Y}(x))\|_1$ to preserve the person's identity.

CycleGAN [28]. Similar to DIAT [20], CycleGAN also learns the mapping between two domains $X \to Y$ and $Y \to X$. To train the domain transfer, it uses a regularization term denoted *cycle consistency loss* combining two cycles: $\|x - G_{Y \to X}(G_{X \to Y}(x))\|_1$ and $\|y - G_{X \to Y}(G_{Y \to X}(y))\|_1$.

IcGAN [26]. Given an input image, IcGAN uses a pretrained encoder-decoder to encode the image into a latent representation in concatenation with an expression vector $\mathbf{y}$ to then reconstruct the original image. It can modify the expression by replacing $\mathbf{y}$ with the desired expression before going through the decoder.

Fig. 7. Sampling the face expression distribution space. As a result of applying our AU-parametrization through the vector $\mathbf{y}_g$, we can synthesize, from the same source image $\mathbf{I}_{\mathbf{y}_r}$, a large variety of photo-realistic images.

StarGAN [4]. An extension of *cycle loss* for simultaneously training between multiple datasets with different data domains. It uses a mask vector to ignore unspecified labels and optimize only on known ground-truth labels. It yields more realistic results when training simultaneously with multiple datasets.

Our model differs from these approaches in two main aspects. First, we do not condition the model on discrete emotions categories, but we learn a basis of anatomically feasible warps that allows generating a continuum of expressions. Secondly, the use of the attention mask allows applying the transformation only on the cropped face, and put it back onto the original image without producing any artifact. As shown in Fig. 6, besides estimating more visually compelling images than other approaches, this results on images of higher spatial resolution.

6.4 High Expressions Variability

Given a single image, we next use our model to produce a wide range of anatomically feasible face expressions while conserving the person's identity. In Fig. 7 all faces are the result of conditioning the input image in the top-left corner with a desired face configuration defined by only 14 AUs. Note the large variability of anatomically feasible expressions that can be synthesized with only 14 AUs.

6.5 Images in the Wild

As previously seen in Fig. 5, the attention mechanism not only learns to focus on specific areas of the face but also allows merging the original and generated image background. This allows our approach to be easily applied to images in the wild while still obtaining high resolution images. For these images we follow the

Fig. 8. Qualitative evaluation on images in the wild. Top: We represent an image (left) from the film "*Pirates of the Caribbean*" and an its generated image obtained by our approach (right). **Bottom:** In a similar manner, we use an image frame (left) from the series "*Game of Thrones*" to synthesize five new images with different expressions.

detection and cropping scheme we described before. Fig. 8 shows two examples on these challenging images. Note how the attention mask allows for a smooth and unnoticeable merge between the entire frame and the generated faces.

6.6 Pushing the Limits of the Model

We next push the limits of our network and discuss the model limitations. We have split success cases into six categories which we summarize in Fig. 9-top. The first two examples (top-row) correspond to human-like sculptures and non-realistic drawings. In both cases, the generator is able to maintain the artistic effects of the original image. Also, note how the attention mask ignores artifacts such as the pixels occluded by the glasses. The third example shows robustness to non-homogeneous textures across the face. Observe that the model is not trying to homogenize the texture by adding/removing the beard's hair. The middle-right category relates to anthropomorphic faces with non-real textures. As for the Avatar image, the network is able to warp the face without affecting its texture. The next category is related to non-standard illuminations/colors for which the model has already been shown robust in Fig. 1. The last and most surprising category is face-sketches (bottom-right). Although the generated face suffers from some artifacts, it is still impressive how the proposed method is still capable of finding sufficient features on the face to transform its expression from worried to excited. The second case shows failures with non-previously seen occlusions such as an eye patch causing artifacts in the missing face attributes.

We have also categorized the failure cases in Fig. 9-bottom, all of them presumably due to insufficient training data. The first case is related to errors in the attention mechanism when given extreme input expressions. The attention does not weight sufficiently the color transformation causing transparencies.

The model also fails when dealing with non-human anthropomorphic distributions as in the case of cyclopes. Lastly, we tested the model behavior when dealing with animals and observed artifacts like human face features.

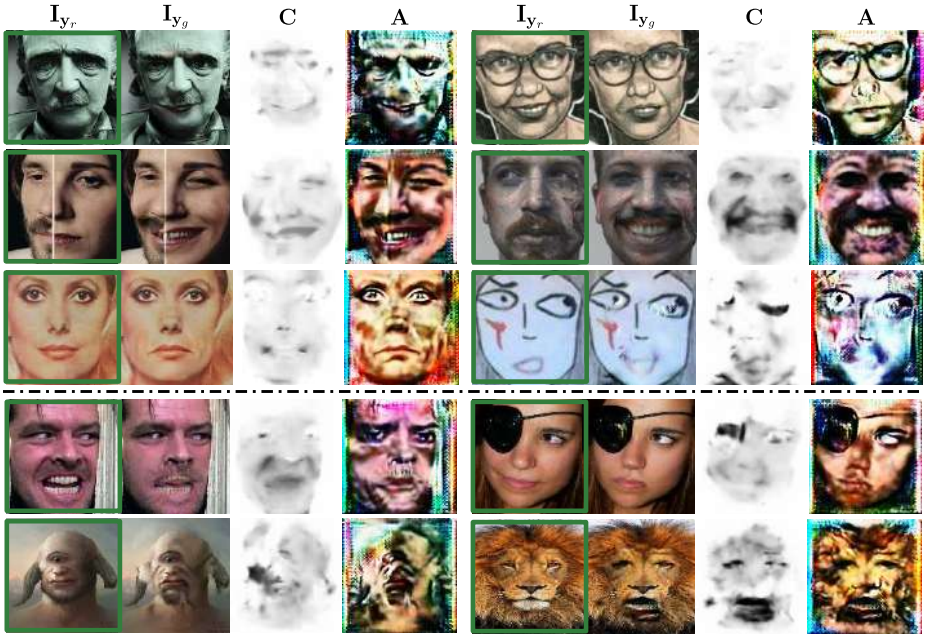


Fig. 9. Success and Failure Cases. In all cases, we represent the source image $\mathbf{I}_{\mathbf{y}_r}$, the target one $\mathbf{I}_{\mathbf{y}_g}$, and the color and attention masks **C** and **A**, respectively. **Top:** Some success cases in extreme situations. **Bottom:** Several failure cases.

7 Conclusions

We have presented a novel GAN model for face animation in the wild that can be trained in a fully unsupervised manner. It advances current works which, so far, had only addressed the problem for discrete emotions category editing and portrait images. Our model encodes anatomically consistent face deformations parameterized by means of AUs. Conditioning the GAN model on these AUs allows the generator to render a wide range of expressions by simple interpolation. Additionally, we embed an attention model within the network which allows focusing only on those regions of the image relevant for every specific expression. By doing this, we can easily process images in the wild, with distracting backgrounds and illumination artifacts. We have exhaustively evaluated the model capabilities and limits in the EmotioNet [3] and RaFD [16] datasets as well as in images from movies. The results are very promising, and show smooth transitions between different expressions. This opens the possibility of applying our approach to video sequences, which we plan to do in the future.

Acknowledgments: This work is partially supported by the Spanish Ministry of Economy and Competitiveness under projects HuMoUR TIN2017-90086-R, ColRobTransp DPI2016-78957 and María de Maeztu Seal of Excellence MDM-2016-0656; by the EU project AEROARMS ICT-2014-1-644271; and by the Grant R01-DC- 014498 of the National Institute of Health. We also thank Nvidia for hardware donation under the GPU Grant Program.

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein GAN. arXiv preprint arXiv:1701.07875 (2017)
2. Baltrušaitis, T., Mahmoud, M., Robinson, P.: Cross-dataset learning and person-specific normalisation for automatic action unit detection. In: FG (2015)
3. Benitez-Quiroz, C.F., Srinivasan, R., Martinez, A.M., et al.: Emotionet: An accurate, real-time algorithm for the automatic annotation of a million facial expressions in the wild. In: CVPR (2016)
4. Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. CVPR (2018)
5. Du, S., Tao, Y., Martinez, A.M.: Compound facial expressions of emotion. Proceedings of the National Academy of Sciences p. 201322355 (2014)
6. Ekman, P., Friesen, W.: Facial action coding system: A technique for the measurement of facial movement. Consulting Psychologists Press (1978)
7. Fischler, M.A., Elschlager, R.A.: The representation and matching of pictorial structures. IEEE Transactions on Computers **22**(1), 67–92 (1973)
8. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS (2014)
9. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein GANs. In: NIPS (2017)
10. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR (2017)
11. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV (2016)
12. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: ICLR (2018)
13. Kim, T., Cha, M., Kim, H., Lee, J., Kim, J.: Learning to discover cross-domain relations with generative adversarial networks. In: ICML (2017)
14. Kingma, D., Ba, J.: ADAM: A method for stochastic optimization. In: ICLR (2015)
15. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
16. Langner, O., Dotsch, R., Bijlstra, G., Wigboldus, D.H., Hawk, S.T., Van Knippenberg, A.: Presentation and validation of the radboud faces database. Cognition and emotion **24**(8), 1377–1388 (2010)
17. Larsen, A.B.L., Sønderby, S.K., Larochelle, H., Winther, O.: Autoencoding beyond pixels using a learned similarity metric. In: ICML (2016)
18. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: CVPR (2017)
19. Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In: ECCV (2016)
20. Li, M., Zuo, W., Zhang, D.: Deep identity-aware transfer of facial attributes. arXiv preprint arXiv:1610.05586 (2016)
21. Liu, M.Y., Breuel, T., Kautz, J.: Unsupervised image-to-image translation networks. In: NIPS (2017)
22. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. In: ICLR (2016)
23. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)

24. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: ICML (2017)
25. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: CVPR (2016)
26. Perarnau, G., van de Weijer, J., Raducanu, B., Álvarez, J.M.: Invertible conditional GANs for image editing. arXiv preprint arXiv:1611.06355 (2016)
27. Pumarola, A., Agudo, A., Sanfeliu, A., Moreno-Noguer, F.: Unsupervised person image synthesis in arbitrary poses. In: CVPR (2018)
28. Radford, A., Metz, L., Chintala, S.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICLR (2016)
29. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee., H.: Generative adversarial text to image synthesis. In: ICML (2016)
30. Scherer, K.R.: Emotion as a process: Function, origin and regulation. Social Science Information **21**, 555–570 (1982)
31. Shen, W., Liu, R.: Learning residual images for face attribute manipulation. In: CVPR (2017)
32. Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., Webb, R.: Learning from simulated and unsupervised images through adversarial training. In: CVPR (2017)
33. Wang, X., Gupta, A.: Generative image modeling using style and structure adversarial networks. In: ECCV (2016)
34. Wang, Z., Liu, D., Yang, J., Han, W., Huang, T.: Deep networks for image super-resolution with sparse prior. In: ICCV (2015)
35. Yu, H., Garrod, O.G., Schyns, P.G.: Perception-driven facial expression synthesis. Computers & Graphics **36**(3) (2012)
36. Zafeiriou, S., Trigeorgis, G., Chrysos, G., Deng, J., Shen, J.: The menpo facial landmark localisation challenge: A step towards the solution. In: CVPRW (2017)
37. Zhang, H., Xu, T., Li, H., Zhang, S., Huang, X., Wang, X., Metaxas, D.: Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: ICCV (2017)
38. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV (2017)
39. Zhu, S., Fidler, S., Urtasun, R., Lin, D., Loy, C.C.: Be your own prada: Fashion synthesis with structural coherence. In: ICCV (2017)

Self-Attention Generative Adversarial Networks

Han Zhang [1 2] **Ian Goodfellow** [3] **Dimitris Metaxas** [1] **Augustus Odena** [2]

Abstract

In this paper, we propose the Self-Attention Generative Adversarial Network (SAGAN) which allows attention-driven, long-range dependency modeling for image generation tasks. Traditional convolutional GANs generate high-resolution details as a function of only spatially local points in lower-resolution feature maps. In SAGAN, details can be generated using cues from all feature locations. Moreover, the discriminator can check that highly detailed features in distant portions of the image are consistent with each other. Furthermore, recent work has shown that generator conditioning affects GAN performance. Leveraging this insight, we apply spectral normalization to the GAN generator and find that this improves training dynamics. The proposed SAGAN performs better than prior work[1], boosting the best published Inception score from 36.8 to 52.52 and reducing Fréchet Inception distance from 27.62 to 18.65 on the challenging ImageNet dataset. Visualization of the attention layers shows that the generator leverages neighborhoods that correspond to object shapes rather than local regions of fixed shape.

1. Introduction

Image synthesis is an important problem in computer vision. There has been remarkable progress in this direction with the emergence of Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), though many open problems remain (Odena, 2019). GANs based on deep convolutional networks (Radford et al., 2016; Karras et al., 2018; Zhang et al.) have been especially successful. However, by carefully examining the generated samples from these models, we can observe that convolutional GANs (Odena et al., 2017; Miyato et al., 2018; Miyato & Koyama, 2018) have much more difficulty in modeling some image classes than others when trained on multi-class datasets (*e.g.*, ImageNet (Russakovsky et al., 2015)). For example, while the state-of-the-art ImageNet GAN model (Miyato & Koyama, 2018) excels at synthesizing image classes with few structural constraints (*e.g.*, ocean, sky and landscape classes, which are distinguished more by texture than by geometry), it fails to capture geometric or structural patterns that occur consistently in some classes (for example, dogs are often drawn with realistic fur texture but without clearly defined separate feet). One possible explanation for this is that previous models rely heavily on convolution to model the dependencies across different image regions. Since the convolution operator has a local receptive field, long range dependencies can only be processed after passing through several convolutional layers. This could prevent learning about long-term dependencies for a variety of reasons: a small model may not be able to represent them, optimization algorithms may have trouble discovering parameter values that carefully coordinate multiple layers to capture these dependencies, and these parameterizations may be statistically brittle and prone to failure when applied to previously unseen inputs. Increasing the size of the convolution kernels can increase the representational capacity of the network but doing so also loses the computational and statistical efficiency obtained by using local convolutional structure. Self-attention (Cheng et al., 2016; Parikh et al., 2016; Vaswani et al., 2017), on the other hand, exhibits a better balance between the ability to model long-range dependencies and the computational and statistical efficiency. The self-attention module calculates response at a position as a weighted sum of the features at all positions, where the weights – or attention vectors – are calculated with only a small computational cost.

In this work, we propose Self-Attention Generative Adversarial Networks (SAGANs), which introduce a self-attention mechanism into convolutional GANs. The self-attention module is complementary to convolutions and helps with modeling long range, multi-level dependencies across image regions. Armed with self-attention, the generator can draw images in which fine details at every location are carefully coordinated with fine details in distant portions of the image. Moreover, the discriminator can also more accurately en-

[1]Department of Computer Science, Rutgers University [2]Google Research, Brain Team [3]Work done while at Google Research. Correspondence to: Han Zhang <zhanghan@google.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]Brock et al. (2018), which builds heavily on this work, has since improved those results substantially.

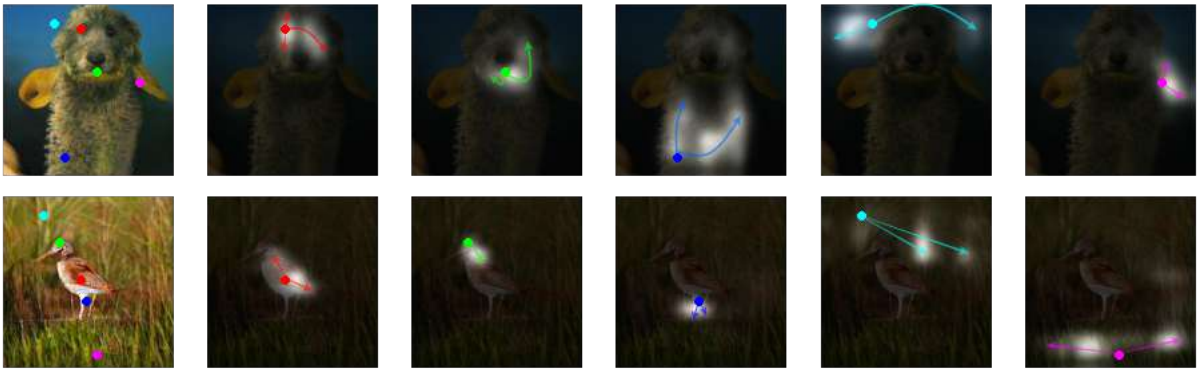


Figure 1. The proposed SAGAN generates images by leveraging complementary features in distant portions of the image rather than local regions of fixed shape to generate consistent objects/scenarios. In each row, the first image shows five representative query locations with color coded dots. The other five images are attention maps for those query locations, with corresponding color coded arrows summarizing the most-attended regions.

force complicated geometric constraints on the global image structure.

In addition to self-attention, we also incorporate recent insights relating network conditioning to GAN performance. The work by (Odena et al., 2018) showed that well-conditioned generators tend to perform better. We propose enforcing good conditioning of GAN generators using the spectral normalization technique that has previously been applied only to the discriminator (Miyato et al., 2018).

We have conducted extensive experiments on the ImageNet dataset to validate the effectiveness of the proposed self-attention mechanism and stabilization techniques. **SAGAN significantly outperforms prior work in image synthesis by boosting the best reported Inception score from 36.8 to 52.52 and reducing Fréchet Inception distance from 27.62 to 18.65.** Visualization of the attention layers shows that the generator leverages neighborhoods that correspond to object shapes rather than local regions of fixed shape. Our code is available at `https://github.com/brain-research/self-attention-gan`.

2. Related Work

Generative Adversarial Networks. GANs have achieved great success in various image generation tasks, including image-to-image translation (Isola et al., 2017; Zhu et al., 2017; Taigman et al., 2017; Liu & Tuzel, 2016; Xue et al., 2018; Park et al., 2019), image super-resolution (Ledig et al., 2017; Snderby et al., 2017) and text-to-image synthesis (Reed et al., 2016b;a; Zhang et al., 2017; Hong et al., 2018). Despite this success, the training of GANs is known to be unstable and sensitive to the choices of hyper-parameters. Several works have attempted to stabilize the GAN training dynamics and improve the sample diversity by designing new network architectures (Radford et al., 2016; Zhang et al., 2017; Karras et al., 2018; 2019), modifying the learning objectives and dynamics (Arjovsky et al., 2017; Salimans et al., 2018; Metz et al., 2017; Che et al., 2017; Zhao et al., 2017; Jolicoeur-Martineau, 2019), adding regularization methods (Gulrajani et al., 2017; Miyato et al., 2018) and introducing heuristic tricks (Salimans et al., 2016; Odena et al., 2017; Azadi et al., 2018). Recently, Miyato *et al.* (Miyato et al., 2018) proposed limiting the spectral norm of the weight matrices in the discriminator in order to constrain the Lipschitz constant of the discriminator function. Combined with the projection-based discriminator (Miyato & Koyama, 2018), the spectrally normalized model greatly improves class-conditional image generation on ImageNet.

Attention Models. Recently, attention mechanisms have become an integral part of models that must capture global dependencies (Bahdanau et al., 2014; Xu et al., 2015; Yang et al., 2016; Gregor et al., 2015; Chen et al., 2018). In particular, self-attention (Cheng et al., 2016; Parikh et al., 2016), also called intra-attention, calculates the response at a position in a sequence by attending to all positions within the same sequence. Vaswani *et al.* (Vaswani et al., 2017) demonstrated that machine translation models could achieve state-of-the-art results by solely using a self-attention model. Parmar *et al.* (Parmar et al., 2018) proposed an Image Transformer model to add self-attention into an autoregressive model for image generation. Wang *et al.* (Wang et al., 2018) formalized self-attention as a non-local operation to model the spatial-temporal dependencies in video sequences. In spite of this progress, self-attention has not yet been explored in the context of GANs. (AttnGAN (Xu et al., 2018) uses attention over word embeddings within an *input* sequence, but not self-attention over *internal model states*). SAGAN learns to efficiently find global, long-range depen-

dencies within internal representations of images.

3. Self-Attention Generative Adversarial Networks

Most GAN-based models (Radford et al., 2016; Salimans et al., 2016; Karras et al., 2018) for image generation are built using convolutional layers. Convolution processes the information in a local neighborhood, thus using convolutional layers alone is computationally inefficient for modeling long-range dependencies in images. In this section, we adapt the non-local model of (Wang et al., 2018) to introduce self-attention to the GAN framework, enabling both the generator and the discriminator to efficiently model relationships between widely separated spatial regions. We call the proposed method Self-Attention Generative Adversarial Networks (SAGAN) because of its self-attention module (see Figure 2).

The image features from the previous hidden layer $\boldsymbol{x} \in \mathbb{R}^{C\times N}$ are first transformed into two feature spaces $\boldsymbol{f}, \boldsymbol{g}$ to calculate the attention, where $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W_f}\boldsymbol{x},\ \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{W_g}\boldsymbol{x}$

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N}\exp(s_{ij})},\ \text{where } s_{ij} = \boldsymbol{f}(\boldsymbol{x_i})^T\boldsymbol{g}(\boldsymbol{x_j}), \quad (1)$$

and $\beta_{j,i}$ indicates the extent to which the model attends to the i^{th} location when synthesizing the j^{th} region. Here, C is the number of channels and N is the number of feature locations of features from the previous hidden layer. The output of the attention layer is $\boldsymbol{o} = (\boldsymbol{o_1}, \boldsymbol{o_2}, ..., \boldsymbol{o_j}, ..., \boldsymbol{o_N}) \in \mathbb{R}^{C\times N}$, where,

$$\boldsymbol{o_j} = \boldsymbol{v}\left(\sum_{i=1}^{N}\beta_{j,i}\boldsymbol{h}(\boldsymbol{x_i})\right),\ \boldsymbol{h}(\boldsymbol{x_i}) = \boldsymbol{W_h}\boldsymbol{x_i},\ \boldsymbol{v}(\boldsymbol{x_i}) = \boldsymbol{W_v}\boldsymbol{x_i}. \quad (2)$$

In the above formulation, $\boldsymbol{W_g} \in \mathbb{R}^{\bar{C}\times C}$, $\boldsymbol{W_f} \in \mathbb{R}^{\bar{C}\times C}$, $\boldsymbol{W_h} \in \mathbb{R}^{\bar{C}\times C}$, and $\boldsymbol{W_v} \in \mathbb{R}^{C\times\bar{C}}$ are the learned weight matrices, which are implemented as 1×1 convolutions. Since We did not notice any significant performance decrease when reducing the channel number of $\bar{C}$ to be C/k, where $k = 1, 2, 4, 8$ after few training epochs on ImageNet. For memory efficiency, we choose $k = 8$ (*i.e.*, $\bar{C} = C/8$) in all our experiments.

In addition, we further multiply the output of the attention layer by a scale parameter and add back the input feature map. Therefore, the final output is given by,

$$\boldsymbol{y_i} = \gamma\boldsymbol{o_i} + \boldsymbol{x_i}, \quad (3)$$

where γ is a learnable scalar and it is initialized as 0. Introducing the learnable γ allows the network to first rely on the cues in the local neighborhood – since this is easier – and then gradually learn to assign more weight to the non-local evidence. The intuition for why we do this is straightforward: we want to learn the easy task first and then progressively increase the complexity of the task. In the SAGAN, the proposed attention module has been applied to both the generator and the discriminator, which are trained in an alternating fashion by minimizing the hinge version of the adversarial loss (Lim & Ye, 2017; Tran et al., 2017; Miyato et al., 2018),

$$\begin{aligned} L_D = &-\mathbb{E}_{(x,y)\sim p_{data}}[\min(0, -1 + D(x, y))] \\ &-\mathbb{E}_{z\sim p_z, y\sim p_{data}}[\min(0, -1 - D(G(z), y))], \quad (4) \\ L_G = &-\mathbb{E}_{z\sim p_z, y\sim p_{data}} D(G(z), y), \end{aligned}$$

4. Techniques to Stabilize the Training of GANs

We also investigate two techniques to stabilize the training of GANs on challenging datasets. First, we use spectral normalization (Miyato et al., 2018) in the generator as well as in the discriminator. Second, we confirm that the two-timescale update rule (TTUR) (Heusel et al., 2017) is effective, and we advocate using it specifically to address slow learning in regularized discriminators.

4.1. Spectral normalization for both generator and discriminator

Miyato *et al.* (Miyato et al., 2018) originally proposed stabilizing the training of GANs by applying spectral normalization to the discriminator network. Doing so constrains the Lipschitz constant of the discriminator by restricting the spectral norm of each layer. Compared to other normalization techniques, spectral normalization does not require extra hyper-parameter tuning (setting the spectral norm of all weight layers to 1 consistently performs well in practice). Moreover, the computational cost is also relatively small.

We argue that the generator can also benefit from spectral normalization, based on recent evidence that the conditioning of the generator is an important causal factor in GANs' performance (Odena et al., 2018). Spectral normalization in the generator can prevent the escalation of parameter magnitudes and avoid unusual gradients. We find empirically that spectral normalization of both generator and discriminator makes it possible to use fewer discriminator updates per generator update, thus significantly reducing the computational cost of training. The approach also shows more stable training behavior.

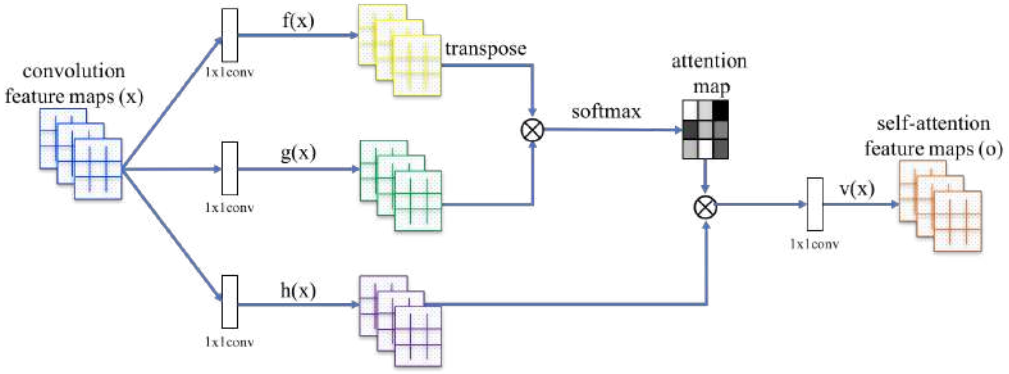


Figure 2. The proposed self-attention module for the SAGAN. The $\otimes$ denotes matrix multiplication. The softmax operation is performed on each row.

4.2. Imbalanced learning rate for generator and discriminator updates

In previous work, regularization of the discriminator (Miyato et al., 2018; Gulrajani et al., 2017) often slows down the GANs' learning process. In practice, methods using regularized discriminators typically require multiple (*e.g.*, 5) discriminator update steps per generator update step during training. Independently, Heusel *et al.* (Heusel et al., 2017) have advocated using separate learning rates (TTUR) for the generator and the discriminator. We propose using TTUR specifically to compensate for the problem of slow learning in a regularized discriminator, making it possible to use fewer discriminator steps per generator step. Using this approach, we are able to produce better results given the same wall-clock time.

5. Experiments

To evaluate the proposed methods, we conducted extensive experiments on the LSVRC2012 (ImageNet) dataset (Russakovsky et al., 2015). First, in Section 5.1, we present experiments designed to evaluate the effectiveness of the two proposed techniques for stabilizing GANs' training. Next, the proposed self-attention mechanism is investigated in Section 5.2. Finally, our SAGAN is compared with state-of-the-art methods (Odena et al., 2017; Miyato & Koyama, 2018) on the image generation task in Section 5.3. Models were trained for roughly 2 weeks on 4 GPUs each, using sychronous SGD (as there are well known difficulties with asynchronous SGD - see e.g. (Odena, 2016)).

Evaluation metrics. We choose the Inception score (IS) (Salimans et al., 2016) and the Fréchet Inception distance (FID) (Heusel et al., 2017) for quantitative evaluation. Though alternatives exist (Zhou et al., 2019; Khrulkov & Oseledets, 2018; Olsson et al., 2018), they are not widely used. The Inception score (Salimans et al., 2016) computes the KL divergence between the conditional class distribution and the marginal class distribution. Higher Inception score indicates better image quality. We include the Inception score because it is widely used and thus makes it possible to compare our results to previous work. However, it is important to understand that Inception score has serious limitations—it is intended primarily to ensure that the model generates samples that can be confidently recognized as belonging to a specific class, and that the model generates samples from many classes, not necessarily to assess realism of details or intra-class diversity. FID is a more principled and comprehensive metric, and has been shown to be more consistent with human evaluation in assessing the realism and variation of the generated samples (Heusel et al., 2017). FID calculates the Wasserstein-2 distance between the generated images and the real images in the feature space of an Inception-v3 network. Besides the FID calculated over the whole data distribution (*i.e.*., all 1000 classes of images in ImageNet), we also compute FID between the generated images and dataset images within each class (called intra FID (Miyato & Koyama, 2018)). Lower FID and intra FID values mean closer distances between synthetic and real data distributions. In all our experiments, 50k samples are randomly generated for each model to compute the Inception score, FID and intra FID.

Network structures and implementation details. All the SAGAN models we train are designed to generate 128×128 images. By default, spectral normalization (Miyato et al., 2018) is used for the layers in both the generator and the discriminator. Similar to (Miyato & Koyama, 2018), SAGAN uses conditional batch normalization in the generator and projection in the discriminator. For all models, we use the Adam optimizer (Kingma & Ba, 2015) with $\beta_1 = 0$ and $\beta_2 = 0.9$ for training. By default, the learning rate for the discriminator is 0.0004 and the learning rate for the generator is 0.0001.

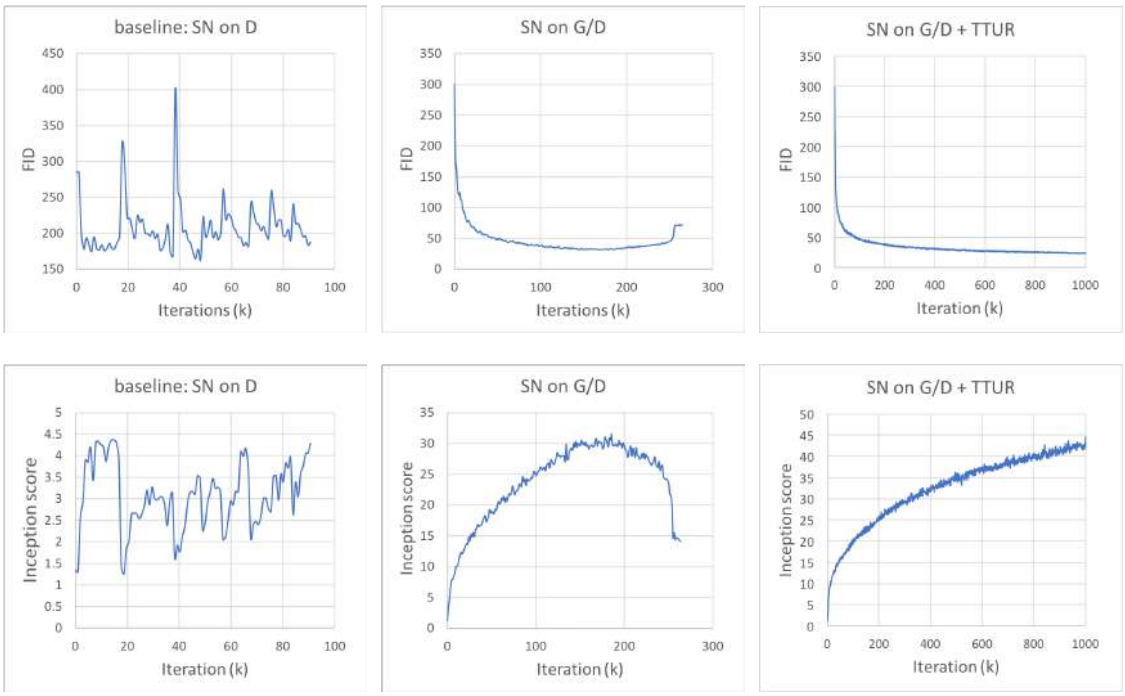


Figure 3. Training curves for the baseline model and our models with the proposed stabilization techniques, "SN on G/D" and two-timescale learning rates (TTUR). All models are trained with 1:1 balanced updates for G and D.

5.1. Evaluating the proposed stabilization techniques

In this section, experiments are conducted to evaluate the effectiveness of the proposed stabilization techniques, *i.e.*, applying spectral normalization (SN) to the generator and utilizing imbalanced learning rates (TTUR). In Figure 3, our models "SN on G/D" and "SN on G/D+TTUR" are compared with a baseline model, which is implemented based on the state-of-the-art image generation method (Miyato et al., 2018). In this baseline model, SN is only utilized in the discriminator. When we train it with 1:1 balanced updates for the discriminator (D) and the generator (G), the training becomes very unstable, as shown in the leftmost sub-figures of Figure 3. It exhibits mode collapse very early in training. For example, the top-left sub-figure of Figure 4 illustrates some images randomly generated by the baseline model at the 10k-th iteration. Although in the the original paper (Miyato et al., 2018) this unstable training behavior is greatly mitigated by using 5:1 imbalanced updates for D and G, the ability to be stably trained with 1:1 balanced updates is desirable for improving the convergence speed of the model. Thus, using our proposed techniques means that the model can produce better results given the same wall-clock time. Given this, there is no need to search for a suitable update ratio for the generator and discriminator. As shown in the middle sub-figures of Figure 3, adding SN to both the generator and the discriminator greatly stabilized our model "SN on G/D", even when it was trained with 1:1 balanced updates. However, the quality of samples does not improve monotonically during training. For example, the image quality as measured by FID and IS is starting to drop at the 260k-th iteration. Example images randomly generated by this model at different iterations can be found in Figure 4. When we also apply the imbalanced learning rates to train the discriminator and the generator, the quality of images generated by our model "SN on G/D+TTUR" improves monotonically during the whole training process. As shown in Figure 3 and Figure 4, we do not observe any significant decrease in sample quality or in the FID or the Inception score during one million training iterations. Thus, both quantitative results and qualitative results demonstrate the effectiveness of the proposed stabilization techniques for GANs' training. They also demonstrate that the effect of the two techniques is at least partly additive. In the rest of experiments, all models use spectral normalization for both the generator and discriminator and use the imbalanced learning rates to train the generator and the discriminator with 1:1 updates.

5.2. Self-attention mechanism.

To explore the effect of the proposed self-attention mechanism, we build several SAGAN models by adding the self-attention mechanism to different stages of the generator and the discriminator. As shown in Table 1, the SAGAN models with the self-attention mechanism at the middle-to-high

Figure 4. 128×128 examples randomly generated by the baseline model and our models "SN on G/D" and "SN on G/D+TTUR".

Model	no attention	SAGAN				Residual			
		$feat_8$	$feat_{16}$	$feat_{32}$	$feat_{64}$	$feat_8$	$feat_{16}$	$feat_{32}$	$feat_{64}$
FID	22.96	22.98	22.14	**18.28**	18.65	42.13	22.40	27.33	28.82
IS	42.87	43.15	45.94	51.43	**52.52**	23.17	44.49	38.50	38.96

Table 1. Comparison of Self-Attention and Residual block on GANs. These blocks are added into different layers of the network. All models have been trained for one million iterations, and the best Inception scores (IS) and Fréchet Inception distance (FID) are reported. $feat_k$ means adding self-attention to the k×k feature maps.

level feature maps (*e.g.*, $feat_{32}$ and $feat_{64}$) achieve better performance than the models with the self-attention mechanism at the low level feature maps (*e.g.*, $feat_8$ and $feat_{16}$). For example, the FID of the model "SAGAN, $feat_8$" is improved from 22.98 to 18.28 by "SAGAN, $feat_{32}$". The reason is that self-attention receives more evidence and enjoys more freedom to choose conditions with larger feature maps (*i.e.*, it is complementary to convolution for large feature maps), however, it plays a similar role as the local convolution when modeling dependencies for small (*e.g.*, 8×8) feature maps. It demonstrates that the attention mechanism gives more power to both the generator and the discriminator to directly model the long-range dependencies in the feature maps. In addition, the comparison of our SAGAN and the baseline model without attention (2nd column of Table 1) further shows the effectiveness of the proposed self-attention mechanism.

Compared with residual blocks with the same number of parameters, the self-attention blocks also achieve better results. For example, the training is not stable when we replace the self-attention block with the residual block in 8×8 feature maps, which leads to a significant decrease in performance (*e.g.*, FID increases from 22.98 to 42.13). Even for the cases when the training goes smoothly, replacing the self-attention block with the residual block still leads to worse results in terms of FID and Inception score. (*e.g.*, FID 18.28 vs 27.33 in feature map 32 × 32). This comparison demonstrates that the performance improvement given by using SAGAN is not simply due to an increase in model depth and capacity.

To better understand what has been learned during the generation process, we visualize the attention weights of the generator in SAGAN for different images. Some sample images with attention are shown in Figure 5 and Figure 1. See the caption of Figure 5 for descriptions of some of the properties of learned attention maps.

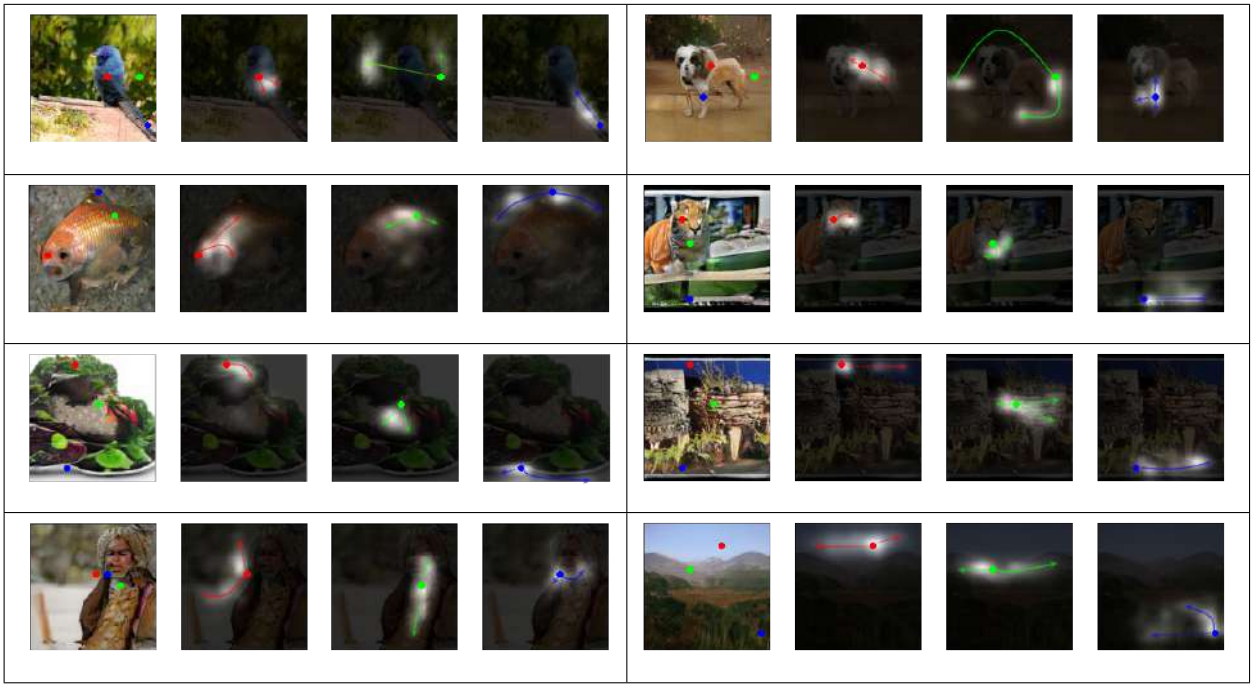


Figure 5. Visualization of attention maps. These images were generated by SAGAN. We visualize the attention maps of the last generator layer that used attention, since this layer is the closest to the output pixels and is the most straightforward to project into pixel space and interpret. In each cell, the first image shows three representative query locations with color coded dots. The other three images are attention maps for those query locations, with corresponding color coded arrows summarizing the most-attended regions. We observe that the network learns to allocate attention according to similarity of color and texture, rather than just spatial adjacency (see the top-left cell). We also find that although some query points are quite close in spatial location, their attention maps can be very different, as shown in the bottom-left cell. As shown in the top-right cell, SAGAN is able to draw dogs with clearly separated legs. The blue query point shows that attention helps to get the structure of the joint area correct. See the text for more discussion about the properties of learned attention maps.

5.3. Comparison with the state-of-the-art

Our SAGAN is also compared with the state-of-the-art GAN models (Odena et al., 2017; Miyato & Koyama, 2018) for class conditional image generation on ImageNet. As shown in Table 2, our proposed SAGAN achieves the best Inception score, intra FID and FID. The proposed SAGAN significantly improves the best published Inception score from 36.8 to 52.52. The lower FID (18.65) and intra FID (83.7) achieved by the SAGAN also indicates that the SAGAN can better approximate the original image distribution by using the self-attention module to model the long-range dependencies between image regions.

Figure 6 shows some comparison results and generated-images for representative classes of ImageNet. We observe that our SAGAN achieves much better performance (*i.e.*, lower intra FID) than the state-of-the-art GAN model (Miyato & Koyama, 2018) for synthesizing image classes with complex geometric or structural patterns, such as goldfish and Saint Bernard. For classes with few structural constraints (*e.g.*, valley, stone wall and coral fungus, which are distinguished more by texture than by geometry), our SAGAN shows less superiority compared with the baseline model (Miyato & Koyama, 2018). Again, the reason is that the self-attention in SAGAN is complementary to the convolution for capturing long-range, global-level dependencies occurring consistently in geometric or structural patterns, but plays a similar role as the local convolution when modeling dependencies for simple texture.

6. Conclusion

In this paper, we proposed Self-Attention Generative Adversarial Networks (SAGANs), which incorporate a self-attention mechanism into the GAN framework. The self-attention module is effective in modeling long-range dependencies. In addition, we show that spectral normalization applied to the generator stabilizes GAN training and that TTUR speeds up training of regularized discriminators. SAGAN achieves the state-of-the-art performance on class-conditional image generation on ImageNet.

Model	Inception Score	Intra FID	FID
AC-GAN (Odena et al., 2017)	28.5	260.0	/
SNGAN-projection (Miyato & Koyama, 2018)	36.8	92.4	27.62*
SAGAN	**52.52**	**83.7**	**18.65**

Table 2. Comparison of the proposed SAGAN with state-of-the-art GAN models (Odena et al., 2017; Miyato & Koyama, 2018) for class conditional image generation on ImageNet. FID of SNGAN-projection is calculated from officially released weights.

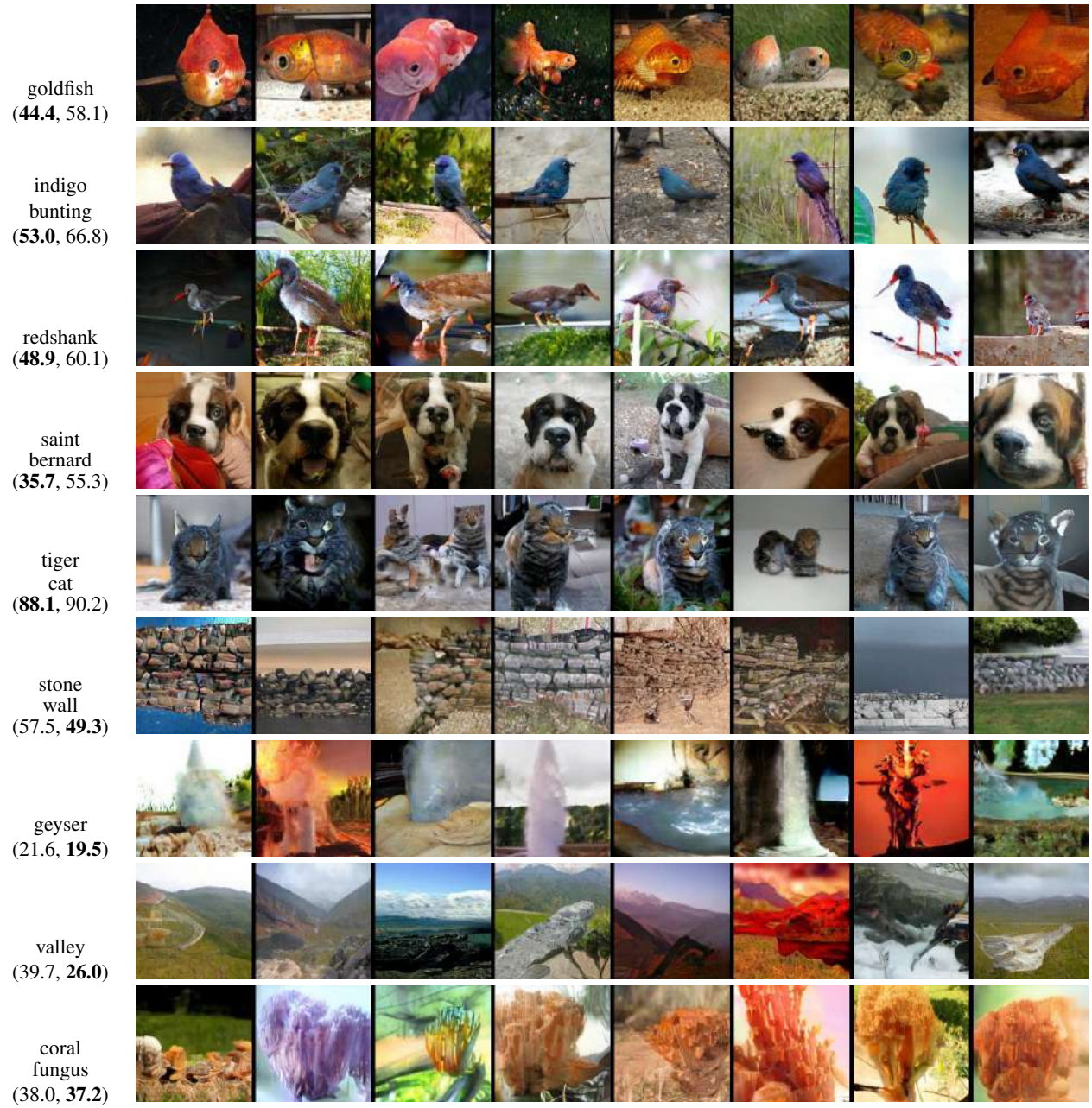


Figure 6. 128x128 example images generated by SAGAN for different classes. Each row shows examples from one class. In the leftmost column, the intra FID of our SAGAN (*left*) and the state-of-the-art method (Miyato & Koyama, 2018)) (*right*) are listed.

Acknowledgments

We thank Surya Bhupatiraju for feedback on drafts of this article. We also thank David Berthelot and Tom B. Brown for help with implementation details. Finally, we thank Jakob Uszkoreit, Tao Xu, and Ashish Vaswani for helpful discussions.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein GAN. *arXiv:1701.07875*, 2017.

Azadi, S., Olsson, C., Darrell, T., Goodfellow, I., and Odena, A. Discriminator rejection sampling. *arXiv preprint arXiv:1810.06758*, 2018.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2014.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Che, T., Li, Y., Jacob, A. P., Bengio, Y., and Li, W. Mode regularized generative adversarial networks. In *ICLR*, 2017.

Chen, X., Mishra, N., Rohaninejad, M., and Abbeel, P. Pixelsnail: An improved autoregressive generative model. In *ICML*, 2018.

Cheng, J., Dong, L., and Lapata, M. Long short-term memory-networks for machine reading. In *EMNLP*, 2016.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial nets. In *NIPS*, 2014.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. DRAW: A recurrent neural network for image generation. In *ICML*, 2015.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein GANs. In *NIPS*, 2017.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, pp. 6629–6640, 2017.

Hong, S., Yang, D., Choi, J., and Lee, H. Inferring semantic layout for hierarchical text-to-image synthesis. In *CVPR*, 2018.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

Jolicoeur-Martineau, A. The relativistic discriminator: a key element missing from standard GAN. In *ICLR*, 2019.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

Khrulkov, V. and Oseledets, I. Geometry score: A method for comparing generative adversarial networks. *arXiv preprint arXiv:1802.02664*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Aitken, A., Tejani, A., Totz, J., Wang, Z., and Shi, W. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

Lim, J. H. and Ye, J. C. Geometric GAN. *arXiv:1705.02894*, 2017.

Liu, M. and Tuzel, O. Coupled generative adversarial networks. In *NIPS*, 2016.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. In *ICLR*, 2017.

Miyato, T. and Koyama, M. cGANs with projection discriminator. In *ICLR*, 2018.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

Odena, A. Faster asynchronous sgd. *arXiv preprint arXiv:1601.04033*, 2016.

Odena, A. Open questions about generative adversarial networks. *Distill*, 2019. doi: 10.23915/distill.00018. https://distill.pub/2019/gan-open-problems.

Odena, A., Olah, C., and Shlens, J. Conditional image synthesis with auxiliary classifier GANs. In *ICML*, 2017.

Odena, A., Buckman, J., Olsson, C., Brown, T. B., Olah, C., Raffel, C., and Goodfellow, I. Is generator conditioning causally related to GAN performance? In *ICML*, 2018.

Olsson, C., Bhupatiraju, S., Brown, T., Odena, A., and Goodfellow, I. Skill rating for generative models. *arXiv preprint arXiv:1808.04888*, 2018.

Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. A decomposable attention model for natural language inference. In *EMNLP*, 2016.

Park, T., Liu, M., Wang, T., and Zhu, J. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

Parmar, N., Vaswani, A., Uszkoreit, J., ukasz Kaiser, Shazeer, N., and Ku, A. Image transformer. *arXiv:1802.05751*, 2018.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

Reed, S., Akata, Z., Mohan, S., Tenka, S., Schiele, B., and Lee, H. Learning what and where to draw. In *NIPS*, 2016a.

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. Generative adversarial text-to-image synthesis. In *ICML*, 2016b.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet large scale visual recognition challenge. *IJCV*, 2015.

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training GANs. In *NIPS*, 2016.

Salimans, T., Zhang, H., Radford, A., and Metaxas, D. N. Improving GANs using optimal transport. In *ICLR*, 2018.

Snderby, C. K., Caballero, J., Theis, L., Shi, W., and Huszar, F. Amortised map inference for image super-resolution. In *ICLR*, 2017.

Taigman, Y., Polyak, A., and Wolf, L. Unsupervised cross-domain image generation. In *ICLR*, 2017.

Tran, D., Ranganath, R., and Blei, D. M. Deep and hierarchical implicit models. *arXiv:1702.08896*, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv:1706.03762*, 2017.

Wang, X., Girshick, R., Gupta, A., and He, K. Non-local neural networks. In *CVPR*, 2018.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X., and He, X. AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, 2018.

Xue, Y., Xu, T., Zhang, H., Long, L. R., and Huang, X. SegAN: Adversarial network with multi-scale L1 loss for medical image segmentation. *Neuroinformatics*, pp. 1–10, 2018.

Yang, Z., He, X., Gao, J., Deng, L., and Smola, A. J. Stacked attention networks for image question answering. In *CVPR*, 2016.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. N. StackGAN++: Realistic image synthesis with stacked generative adversarial networks. *TPAMI*.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

Zhao, J., Mathieu, M., and LeCun, Y. Energy-based generative adversarial network. In *ICLR*, 2017.

Zhou, S., Gordon, M., Krishna, R., Narcomey, A., Morina, D., and Bernstein, M. S. HYPE: human eye perceptual evaluation of generative models. *CoRR*, abs/1904.01121, 2019. URL http://arxiv.org/abs/1904.01121.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

Video-to-Video Synthesis

Ting-Chun Wang[1], **Ming-Yu Liu**[1], **Jun-Yan Zhu**[2], **Guilin Liu**[1],
Andrew Tao[1], **Jan Kautz**[1], **Bryan Catanzaro**[1]
[1]NVIDIA, [2]MIT CSAIL
{tingchunw,mingyul,guilinl,atao,jkautz,bcatanzaro}@nvidia.com,
junyanz@mit.edu

Abstract

We study the problem of video-to-video synthesis, whose goal is to learn a mapping function from an input source video (e.g., a sequence of semantic segmentation masks) to an output photorealistic video that precisely depicts the content of the source video. While its image counterpart, the image-to-image translation problem, is a popular topic, the video-to-video synthesis problem is less explored in the literature. Without modeling temporal dynamics, directly applying existing image synthesis approaches to an input video often results in temporally incoherent videos of low visual quality. In this paper, we propose a video-to-video synthesis approach under the generative adversarial learning framework. Through carefully-designed generators and discriminators, coupled with a spatio-temporal adversarial objective, we achieve high-resolution, photorealistic, temporally coherent video results on a diverse set of input formats including segmentation masks, sketches, and poses. Experiments on multiple benchmarks show the advantage of our method compared to strong baselines. In particular, our model is capable of synthesizing 2K resolution videos of street scenes up to 30 seconds long, which significantly advances the state-of-the-art of video synthesis. Finally, we apply our method to future video prediction, outperforming several competing systems. Code, models, and more results are available at our website.

1 Introduction

The capability to model and recreate the dynamics of our visual world is essential to building intelligent agents. Apart from purely scientific interests, learning to synthesize continuous visual experiences has a wide range of applications in computer vision, robotics, and computer graphics. For example, in model-based reinforcement learning [2, 24], a video synthesis model finds use in approximating visual dynamics of the world for training the agent with less amount of real experience data. Using a learned video synthesis model, one can generate realistic videos without explicitly specifying scene geometry, materials, lighting, and dynamics, which would be cumbersome but necessary when using a standard graphics rendering engine [35].

The video synthesis problem exists in various forms, including future video prediction [15, 18, 42, 45, 50, 65, 68, 71, 77] and unconditional video synthesis [59, 67, 69]. In this paper, we study a new form: *video-to-video synthesis*. At the core, we aim to learn a mapping function that can convert an input video to an output video. To the best of our knowledge, a general-purpose solution to video-to-video synthesis has not yet been explored by prior work, although its image counterpart, the image-to-image translation problem, is a popular research topic [6, 31, 33, 43, 44, 63, 66, 73, 82, 83]. Our method is inspired by previous application-specific video synthesis methods [58, 60, 61, 75].

We cast the video-to-video synthesis problem as a distribution matching problem, where the goal is to train a model such that the conditional distribution of the synthesized videos given input videos resembles that of real videos. To this end, we learn a conditional generative adversarial model [20]

Preprint. Work in progress.

Figure 1: Generating a photorealistic video from an input segmentation map video on Cityscapes. Top left: input. Top right: `pix2pixHD`. Bottom left: `COVST`. Bottom right: `vid2vid` (ours). *Click the image to play the video clip in a browser.*

given paired input and output videos, With carefully-designed generators and discriminators, and a new spatio-temporal learning objective, our method can learn to synthesize high-resolution, photorealistic, temporally coherent videos. Moreover, we extend our method to multimodal video synthesis. Conditioning on the same input, our model can produce videos with diverse appearances.

We conduct extensive experiments on several datasets on the task of converting a sequence of segmentation masks to photorealistic videos. Both quantitative and qualitative results indicate that our synthesized footage looks more photorealistic than those from strong baselines. See Figure 1 for example. We further demonstrate that the proposed approach can generate photorealistic 2K resolution videos, up to 30 seconds long. Our method also grants users flexible high-level control over the video generation results. For example, a user can easily replace all the buildings with trees in a street view video. In addition, our method works for other input video formats such as face sketches and body poses, enabling many applications from face swapping to human motion transfer. Finally, we extend our approach to future prediction and show that our method can outperform existing systems. Please visit our website for code, models, and more results.

2 Related Work

Generative Adversarial Networks (GANs). We build our model on GANs [20]. During GAN training, a generator and a discriminator play a zero-sum game. The generator aims to produce realistic synthetic data so that the discriminator cannot differentiate between real and the synthesized data. In addition to noise distributions [14, 20, 55], various forms of data can be used as input to the generator, including images [33, 43, 82], categorical labels [52, 53], and textual descriptions [56, 80]. Such conditional models are called conditional GANs, and allow flexible control over the output of the model. Our method belongs to the category of conditional video generation with GANs. However, instead of predicting future videos conditioning on the current observed frames [41, 50, 69], our method synthesizes photorealistic videos conditioning on manipulable semantic representations, such as segmentation masks, sketches, and poses.

Image-to-image translation algorithms transfer an input image from one domain to a corresponding image in another domain. There exists a large body of work for this problem [6, 31, 33, 43, 44, 63, 66, 73, 82, 83]. Our approach is their video counterpart. In addition to ensuring that each video frame looks photorealistic, a video synthesis model also has to produce temporally coherent frames, which is a challenging task, especially for a long duration video.

Unconditional video synthesis. Recent work [59, 67, 69] extends the GAN framework for unconditional video synthesis, which learns a generator for converting a random vector to a video.

VGAN [69] uses a spatio-temporal convolutional network. TGAN [59] projects a latent code to a set of latent image codes and uses an image generator to convert those latent image codes to frames. MoCoGAN [67] disentangles the latent space to motion and content subspaces and uses a recurrent neural network to generate a sequence of motion codes. Due to the unconditional setting, these methods often produce low-resolution and short-length videos.

Future video prediction. Conditioning on the observed frames, video prediction models are trained to predict future frames [15, 18, 36, 41, 42, 45, 50, 65, 68, 71, 72, 77]. Many of these models are trained with image reconstruction losses, often producing blurry videos due to the classic regress-to-the-mean problem. Also, they fail to generate long duration videos even with adversarial training [42, 50]. The video-to-video synthesis problem is substantially different because it does not attempt to predict object motions or camera motions. Instead, our approach is conditional on an existing video and can produce high-resolution and long-length videos in a different domain.

Video-to-video synthesis. While video super-resolution [61, 62], video matting and blending [3, 12], and video inpainting [74] can be considered as special cases of the video-to-video synthesis problem, existing approaches rely on problem-specific constraints and designs. Hence, these methods cannot be easily applied to other applications. Video style transfer [10, 22, 28, 58], transferring the style of a reference painting to a natural scene video, is also related. In Section 4 , we show that our method outperforms a strong baseline that combines a recent video style transfer with a state-of-the-art image-to-image translation approach.

3 Video-to-Video Synthesis

Let $\mathbf{s}_1^T \equiv \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T\}$ be a sequence of source video frames. For example, it can be a sequence of semantic segmentation masks or edge maps. Let $\mathbf{x}_1^T \equiv \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$ be the sequence of corresponding real video frames. The goal of video-to-video synthesis is to learn a mapping function that can convert $\mathbf{s}_1^T$ to a sequence of output video frames, $\tilde{\mathbf{x}}_1^T \equiv \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_T\}$, so that the conditional distribution of $\tilde{\mathbf{x}}_1^T$ given $\mathbf{s}_1^T$ is identical to the conditional distribution of $\mathbf{x}_1^T$ given $\mathbf{s}_1^T$.

$$p(\tilde{\mathbf{x}}_1^T|\mathbf{s}_1^T) = p(\mathbf{x}_1^T|\mathbf{s}_1^T). \tag{1}$$

Through matching the conditional video distributions, the model learns to generate photorealistic, temporally coherent output sequences as if they were captured by a video camera.

We propose a conditional GAN framework for this conditional video distribution matching task. Let G be a generator that maps an input source sequence to a corresponding output frame sequence: $\mathbf{x}_1^T = G(\mathbf{s}_1^T)$. We train the generator by solving the minimax optimization problem given by

$$\max_D \min_G E_{(\mathbf{x}_1^T, \mathbf{s}_1^T)}[\log D(\mathbf{x}_1^T, \mathbf{s}_1^T)] + E_{\mathbf{s}_1^T}[\log(1 - D(G(\mathbf{s}_1^T), \mathbf{s}_1^T))], \tag{2}$$

where D is the discriminator. We note that as solving (2), we minimize the Jensen-Shannon divergence between $p(\tilde{\mathbf{x}}_1^T|\mathbf{s}_1^T)$ and $p(\mathbf{x}_1^T|\mathbf{s}_1^T)$ as shown by Goodfellow et al. [20].

Solving the minimax optimization problem in (2) is a well-known, challenging task. Careful designs of network architectures and objective functions are essential to achieve good performance as shown in the literature [14, 21, 30, 37, 49, 51, 55, 73, 80]. We follow the same spirit and propose new network designs and a spatio-temporal objective for video-to-video synthesis as detailed below.

Sequential generator. To simplify the video-to-video synthesis problem, we make a Markov assumption where we factorize the conditional distribution $p(\tilde{\mathbf{x}}_1^T|\mathbf{s}_1^T)$ to a product form given by

$$p(\tilde{\mathbf{x}}_1^T|\mathbf{s}_1^T) = \prod_{t=1}^{T} p(\tilde{\mathbf{x}}_t|\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^t). \tag{3}$$

In other words, we assume the video frames can be generated sequentially, and the generation of the t-th frame $\tilde{\mathbf{x}}_t$ only depends on three factors: 1) current source frame $\mathbf{s}_t$, 2) past L source frames $\mathbf{s}_{t-L}^{t-1}$, and 3) past L generated frames $\tilde{\mathbf{x}}_{t-L}^{t-1}$. We train a feed-forward network F to model the conditional distribution $p(\tilde{\mathbf{x}}_t|\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^t)$ using $\tilde{\mathbf{x}}_t = F(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^t)$. We obtain the final output $\tilde{\mathbf{x}}_1^T$ by applying the function F in a recursive manner. We found that a small L (e.g., $L = 1$) causes training instability, while a large L increases training time and GPU memory but with minimal quality improvement. In our experiments, we set $L = 2$.

Video signals contain a large amount of redundant information in consecutive frames. If the optical flow [46] between consecutive frames is known, we can estimate the next frame by warping the current frame [54, 70]. This estimation would be largely correct except for the occluded areas. Based on this observation, we model F as

$$F(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^{t}) = (\mathbf{1} - \tilde{\mathbf{m}}_t) \odot \tilde{\mathbf{w}}_{t-1}(\tilde{\mathbf{x}}_{t-1}) + \tilde{\mathbf{m}}_t \odot \tilde{\mathbf{h}}_t, \tag{4}$$

where $\odot$ is the element-wise product operator and $\mathbf{1}$ is an image of all ones. The first part corresponds to pixels warped from the previous frame, while the second part hallucinates new pixels. The definitions of the other terms in Equation 4 are given below.

- $\tilde{\mathbf{w}}_{t-1} = W(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^{t})$ is the estimated optical flow from $\tilde{\mathbf{x}}_{t-1}$ to $\tilde{\mathbf{x}}_t$, and W is the optical flow prediction network. We estimate the optical flow using both input source images $\mathbf{s}_{t-L}^{t}$ and previously synthesized images $\tilde{\mathbf{x}}_{t-L}^{t-1}$. By $\tilde{\mathbf{w}}_{t-1}(\tilde{\mathbf{x}}_{t-1})$, we warp $\tilde{\mathbf{x}}_{t-1}$ based on $\tilde{\mathbf{w}}_{t-1}$.
- $\tilde{\mathbf{h}}_t = H(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^{t})$ is the hallucinated image, synthesized directly by the generator H.
- $\tilde{\mathbf{m}}_t = M(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^{t})$ is the occlusion mask with continuous values between 0 and 1. M denotes the mask prediction network. Our occlusion mask is soft instead of binary to better handle the "zoom in" scenario. For example, when an object is moving closer to our camera, the object will become blurrier over time if we only warp previous frames. To increase the resolution of the object, we need to synthesize new texture details. By using a soft mask, we can add details by gradually blending the warped pixels and the newly synthesized pixels.

We use residual networks [26] for M, W, and H. To generate high-resolution videos, we adopt a coarse-to-fine generator design similar to the method of Wang et. al [73].

As using multiple discriminators can mitigate the mode collapse problem during GANs training [19, 67, 73], we also design two types of discriminators as detailed below.

Conditional image discriminator D_I. The purpose of D_I is to ensure that each output frame resembles a real image given the same source image. This conditional discriminator should output 1 for a true pair $(\mathbf{x}_t, \mathbf{s}_t)$ and 0 for a fake one $(\tilde{\mathbf{x}}_t, \mathbf{s}_t)$.

Conditional video discriminator D_V. The purpose of D_V is to ensure that consecutive output frames resemble the temporal dynamics of a real video given the same optical flow. While D_I conditions on the source image, D_V conditions on the flow. Let $\mathbf{w}_{t-K}^{t-2}$ be $K-1$ optical flow for the K consecutive real images $\mathbf{x}_{t-K}^{t-1}$. This conditional discriminator D_V should output 1 for a true pair $(\mathbf{x}_{t-K}^{t-1}, \mathbf{w}_{t-K}^{t-2})$ and 0 for a fake one $(\tilde{\mathbf{x}}_{t-K}^{t-1}, \mathbf{w}_{t-K}^{t-2})$.

We introduce two sampling operators to facilitate the discussion. First, let ϕ_I be a random image sampling operator such that $\phi_I(\mathbf{x}_1^T, \mathbf{s}_1^T) = (\mathbf{x}_i, \mathbf{s}_i)$ where i is an integer uniformly sampled from 1 to T. In other words, ϕ_I randomly samples a pair of images from $(\mathbf{x}_1^T, \mathbf{s}_1^T)$. Second, we define ϕ_V as a sampling operator that randomly retrieve K consecutive frames. Specifically, $\phi_V(\mathbf{w}_1^{T-1}, \mathbf{x}_1^T, \mathbf{s}_1^T) = (\mathbf{w}_{i-K}^{i-2}, \mathbf{x}_{i-K}^{i-1}, \mathbf{s}_{i-K}^{i-1})$ where i is an integer uniformly sampled from $K+1$ to $T+1$. This operator retrieves K consecutive frames and the corresponding $K-1$ optical flow images. With ϕ_I and ϕ_V, we are ready to present our learning objective function.

Learning objective function. We train the sequential video synthesis function F by solving

$$\min_F \big(\max_{D_I} \mathcal{L}_I(F, D_I) + \max_{D_V} \mathcal{L}_V(F, D_V) \big) + \lambda_W \mathcal{L}_W(F), \tag{5}$$

where $\mathcal{L}_I$ is the GAN loss on images defined by the conditional image discriminator D_I, $\mathcal{L}_V$ is the GAN loss on K consecutive frames defined by D_V, and $\mathcal{L}_W$ is the flow estimation loss. The weight λ_W is set to 10 throughout the experiments based on a grid search. In addition to the loss terms in Equation 5, we use the discriminator feature matching loss [40, 73] and VGG feature matching loss [16, 34, 73] as they improve the convergence speed and training stability [73]. Please see the appendix for more details.

We further define the image-conditional GAN loss $\mathcal{L}_I$ [33] using the operator ϕ_I

$$E_{\phi_I(\mathbf{x}_1^T, \mathbf{s}_1^T)}[\log D_I(\mathbf{x}_i, \mathbf{s}_i)] + E_{\phi_I(\tilde{\mathbf{x}}_1^T, \mathbf{s}_1^T)}[\log(1 - D_I(\tilde{\mathbf{x}}_i, \mathbf{s}_i))]. \tag{6}$$

Similarly, the video GAN loss $\mathcal{L}_V$ is given by

$$E_{\phi_V(\mathbf{w}_1^{T-1}, \mathbf{x}_1^T, \mathbf{s}_1^T)}[\log D_V(\mathbf{x}_{i-K}^{i-1}, \mathbf{w}_{i-K}^{i-2})] + E_{\phi_V(\mathbf{w}_1^{T-1}, \tilde{\mathbf{x}}_1^T, \mathbf{s}_1^T)}[\log(1 - D_V(\tilde{\mathbf{x}}_{i-K}^{i-1}, \mathbf{w}_{i-K}^{i-2}))]. \tag{7}$$

Recall that we synthesize a video $\tilde{\mathbf{x}}_1^T$ by recursively applying F.

The flow loss $\mathcal{L}_W$ includes two terms. The first is the endpoint error between the ground truth and the estimated flow, and the second is the warping loss when the flow warps the previous frame to the next frame. Let $\mathbf{w}_t$ be the ground truth flow from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$. The flow loss $\mathcal{L}_W$ is given by

$$\mathcal{L}_W = \frac{1}{T-1}\sum_{t=1}^{T-1}\big(||\tilde{\mathbf{w}}_t - \mathbf{w}_t||_1 + \|\tilde{\mathbf{w}}_t(\mathbf{x}_t) - \mathbf{x}_{t+1}\|_1\big). \tag{8}$$

Foreground-background prior. When using semantic segmentation masks as the source video, we can divide an image into foreground and background areas based on the semantics. For example, buildings and roads belong to the background, while cars and pedestrians are considered as the foreground. We leverage this strong foreground-background prior in the generator design to further improve the synthesis performance of the proposed model.

In particular, we decompose the image hallucination network H into a foreground model $\tilde{\mathbf{h}}_{F,t} = H_F(\mathbf{s}_{t-L}^t)$ and a background model $\tilde{\mathbf{h}}_{B,t} = H_B(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^t)$. We note that background motion can be modeled as a global transformation in general, where optical flow can be estimated quite accurately. As a result, the background region can be generated accurately via warping, and the background hallucination network H_B only needs to synthesize the occluded areas. On the other hand, a foreground object often has a large motion and only occupies a small portion of the image, which makes optical flow estimation difficult. The network H_F has to synthesize most of the foreground content from scratch. With this foreground–background prior, F is then given by

$$F(\tilde{\mathbf{x}}_{t-L}^{t-1}, \mathbf{s}_{t-L}^t) = (\mathbf{1} - \tilde{\mathbf{m}}_t) \odot \tilde{\mathbf{w}}_{t-1}(\tilde{\mathbf{x}}_{t-1}) + \tilde{\mathbf{m}}_t \odot \big((\mathbf{1} - \mathbf{m}_{B,t}) \odot \tilde{\mathbf{h}}_{F,t} + \mathbf{m}_{B,t} \odot \tilde{\mathbf{h}}_{B,t}\big), \tag{9}$$

where $\mathbf{m}_{B,t}$ is the background mask derived from the ground truth segmentation mask $\mathbf{s}_t$. This prior improves the visual quality by a large margin with the cost of minor flickering artifacts. In Table 2, our user study shows that most people prefer the results with foreground–background modeling.

Multimodal synthesis. The synthesis network F is a unimodal mapping function. Given an input source video, it can only generate one output video. To achieve multimodal synthesis [19, 73, 83], we adopt a feature embedding scheme [73] for the source video that consists of instance-level semantic segmentation masks. Specifically, at training time, we train an image encoder E to encode the ground truth real image $\mathbf{x}_t$ into a d-dimensional feature map ($d = 3$ in our experiments). We then apply an instance-wise average pooling to the map so that all the pixels within the same object share the same feature vectors. We then feed both the instance-wise averaged feature map $\mathbf{z}_t$ and the input semantic segmentation mask $\mathbf{s}_t$ to the generator F. Once training is done, we fit a mixture of Gaussian distribution to the feature vectors that belong to the same object class. At test time, we sample a feature vector for each object instance using the estimated distribution of that object class. Given different feature vectors, the generator F can synthesize videos with different visual appearances.

4 Experiments

Implementation details. We train our network in a spatio-temporally progressive manner. In particular, we start with generating low-resolution videos with few frames, and all the way up to generating full resolution videos with 30 (or more) frames. Our coarse-to-fine generator consists of three scales: 512×256, 1024×512, and 2048×1024 resolutions, respectively. The mask prediction network M and flow prediction network W share all the weights except for the output layer. We use the multi-scale PatchGAN discriminator architecture [33, 73] for the image discriminator D_I. In addition to multi-scale in the spatial resolution, our multi-scale video discriminator D_V also looks at different frame rates of the video to ensure both short-term and long-term consistency. See the appendix for more details.

We train our model for 40 epochs using the ADAM optimizer [39] with lr $= 0.0002$ and $(\beta_1, \beta_2) = (0.5, 0.999)$ on an NVIDIA DGX1 machine. We use the LSGAN loss [49]. Due to the high image resolution, even with one short video per batch, we have to use all the GPUs in DGX1 (8 V100 GPUs, each with 16GB memory) for training. We distribute the generator computation task to 4 GPUs and the discriminator task to the other 4 GPUs. Training takes ~ 10 days for 2K resolution.

Datasets. We evaluate the proposed approach on several datasets.

Table 1: Comparison between competing video-to-video synthesis approaches on Cityscapes.

Fréchet Inception Dist.	I3D	ResNeXt
pix2pixHD	5.57	0.18
COVST	5.55	0.18
vid2vid (ours)	**4.66**	**0.15**

Human Preference Score	short seq.	long seq.
vid2vid (ours) / pix2pixHD	**0.87** / 0.13	**0.83** / 0.17
vid2vid (ours) / COVST	**0.84** / 0.16	**0.80** / 0.20

Table 2: Ablation study. We compare the proposed approach to its three variants.

Human Preference Score	
vid2vid (ours) / no background-foreground prior	**0.80** / 0.20
vid2vid (ours) / no conditional video discriminator	**0.84** / 0.16
vid2vid (ours) / no flow warping	**0.67** / 0.33

Table 3: Comparison between future video prediction methods on Cityscapes.

Fréchet Inception Dist.	I3D	ResNeXt
PredNet	11.18	0.59
MCNet	10.00	0.43
vid2vid (ours)	**3.44**	**0.18**

Human Preference Score	
vid2vid (ours) / PredNet	**0.92** / 0.08
vid2vid (ours) / MCNet	**0.98** / 0.02

- **Cityscapes** [13]. The dataset consists of 2048×1024 street scene videos captured in several German cities. Only a subset of images in the videos contains ground truth semantic segmentation masks. To obtain the input source videos, we use those images to train a DeepLabV3 semantic segmentation network [11] and apply the trained network to segment all the videos. We use the optical flow extracted by FlowNet2 [32] as the ground truth flow $\mathbf{w}$. We treat the instance segmentation masks computed by the Mask R-CNN [25] as our instance-level ground truth. In summary, the training set contains 2975 videos, each with 30 frames. The validation set consists of 500 videos, each with 30 frames. Finally, we test our method on three long sequences from the Cityscapes demo videos, with 600, 1100, and 1200 frames, respectively. We will show that although trained on short videos, our model can synthesize long videos.
- **Apolloscape** [29] consists of 73 street scene videos captured in Beijing, whose video lengths vary from 100 to 1000 frames. Similar to Cityscapes, Apolloscape is constructed for the image/video semantic segmentation task. But we use it for synthesizing videos using the semantic segmentation mask. We split the dataset into half for training and validation.
- **Face video dataset** [57]. We use the real videos in the FaceForensics dataset, which contains 854 videos of news briefing from different reporters. We use this dataset for the sketch video to face video synthesis task. To extract a sequence of sketches from a video, we first apply a face alignment algorithm [38] to localize facial landmarks in each frame. The facial landmarks are then connected to create the face sketch. For background, we extract Canny edges outside the face regions. We split the dataset into 704 videos for training and 150 videos for validation.
- **Dance video dataset.** We download YouTube dance videos for the pose to human motion synthesis task. Each video is about $3 \sim 4$ minutes long at 1280×720 resolution, and we crop the central 512×720 regions. We extract human poses with DensePose [23] and OpenPose [7], and directly concatenate the results together. Each training set includes a dance video from a single dancer, while the test set contains videos of other dance motions or from other dancers.

Baselines. We compare our approach to two baselines trained on the same data.

- pix2pixHD [73] is the state-of-the-art image-to-image translation approach. When applying the approach to the video-to-video synthesis task, we process input videos frame-by-frame.
- COVST is built on the coherent video style transfer [10] by replacing the stylization network with pix2pixHD. The key idea in COVST is to warp high-level deep features using optical flow for achieving temporally coherent outputs. No additional adversarial training is applied. We feed in ground truth optical flow to COVST, which is impractical for real applications. In contrast, our model estimates optical flow from source videos.

Evaluation metrics. We use both subjective and objective metrics for evaluation.

- **Human preference score.** We perform a human subjective test for evaluating the visual quality of synthesized videos. We use the Amazon Mechanical Turk (AMT) platform. During each

Figure 2: Apolloscape results. Left: `pix2pixHD`. Center: `COVST`. Right: proposed. The input semantic segmentation mask video is shown in the left video. *Click the image to play the video clip in a browser.*

Figure 3: Example multi-modal video synthesis results. These synthesized videos contain different road surfaces. *Click the image to play the video clip in a browser.*

Figure 4: Example results of changing input semantic segmentation masks to generate diverse videos. Left: tree→building. Right: building→tree. The original video is shown in Figure 3. *Click the image to play the video clip in a browser.*

test, an AMT participant is first shown two videos at a time (results synthesized by two different algorithms) and then asked which one looks more like a video captured by a real camera. We specifically ask the worker to check for both temporal coherence and image quality. A worker must have a life-time task approval rate greater than 98% to participate in the evaluation. For each question, we gather answers from 10 different workers. We evaluate the algorithm by the ratio that the algorithm outputs are preferred.

- **Fréchet Inception Distance (FID)** [27] is a widely used metric for implicit generative models, as it correlates well with the visual quality of generated samples. The FID was originally developed for evaluating image generation. We propose a variant for video evaluation, which measures both visual quality and temporal consistency. Specifically, we use a pre-trained video recognition CNN as a feature extractor after removing the last few layers from the network. This feature extractor will be our "inception" network. For each video, we extract a spatio-temporal feature map with this CNN. We then compute the mean $\tilde{\mu}$ and covariance matrix $\tilde{\Sigma}$ for the feature vectors from all the synthesized videos. We also calculate the same quantities μ and Σ for the ground truth videos. The FID is then calculated as $\|\mu - \tilde{\mu}\|^2 + \text{Tr}\Big(\Sigma + \tilde{\Sigma} - 2\sqrt{\Sigma\tilde{\Sigma}}\Big)$. We use two different pre-trained video recognition CNNs in our evaluation: I3D [8] and ResNeXt [76].

Main results. We compare the proposed approach to the baselines on the Cityscapes benchmark, where we apply the learned models to synthesize 500 short video clips in the validation set. As shown in Table 1, our results have a smaller FID and are often favored by the human subjects. We also report the human preference scores on the three long test videos. Again, the videos rendered by our approach are considered more realistic by the human subjects. The human preference scores for the Apolloscape dataset are given in the appendix.

Figure 5: Example face→sketch→face results. Each set shows the original video, the extracted edges, and our synthesized video. *Click the image to play the video clip in a browser.*

Figure 6: Example dance→pose→dance results. Each set shows the original dancer, the extracted poses, and the synthesized video. *Click the image to play the video clip in a browser.*

Figures 1 and 2 show the video synthesis results. Although each frame rendered by `pix2pixHD` is photorealistic, the resulting video lacks temporal coherence. The road lane markings and building appearances are inconsistent across frames. While improving upon `pix2pixHD`, `COVST` still suffers from temporal inconsistency. On the contrary, our approach produces a high-resolution, photorealistic, temporally consistent video output. We can generate 30-second long videos, showing that our approach synthesizes convincing videos with longer lengths.

We conduct an ablation study to analyze several design choices of our method. Specifically, we create three variants. In one variant, we do not use the foreground-background prior, which is termed `no background-foreground prior`. That is, instead of using Equation 9, we use Equation 4. The second variant is `no conditional video discriminator` where we do not use the video discriminator D_V for training. In the last variant, we remove the optical flow prediction network W and the mask prediction network M from the generator F in Equation 4 and only use H for synthesis. This variant is referred to as `no flow warping`. We use the human preference score on Cityscapes for this ablation study. Table 2 shows that the visual quality of output videos degrades significantly without the ablated components. To evaluate the effectiveness of different components in our network, we also experimented with directly using ground truth flows instead of estimated flows by our network. We found the results visually similar, which suggests that our network is robust to the errors in the estimated flows.

Multimodal results. Figure 3 shows example multimodal synthesis results. In this example, we keep the sampled feature vectors of all the object instances in the video the same except for the road instance. The figure shows temporally smooth videos with different road appearances.

Semantic manipulation. Our approach also allows the user to manipulate the semantics of source videos. In Figure 4, we show an example of changing the semantic labels. In the left video, we replace all trees with buildings in the original segmentation masks and synthesize a new video. On the right, we show the result of replacing buildings with trees.

Sketch-to-video synthesis for face swapping. We train a sketch-to-face synthesis video model using the real face videos in the FaceForensics dataset [57]. As shown in Figure 5, our model can convert sequences of sketches to photorealistic output videos. This model can be used to change the facial appearance of the original face videos [5].

Pose-to-video synthesis for human motion transfer. We also apply our method to the task of converting sequences of human poses to photorealistic output videos. We note that the image counterpart was studied in recent works [4, 17, 47, 48]. As shown in Figure 6, our model learns to synthesize high-resolution photorealistic output dance videos that contain unseen body shapes and motions. Our method can change the clothing [79, 81] for the same dancer (Figure 6 left) as well as transfer the visual appearance to new dancers (Figure 6 right) as explored in concurrent work [1, 9, 78].

Figure 7: Future video prediction results. Top left: ground truth. Top right: PredNet [45]. Bottom left: MCNet [68]. Bottom right: ours. *Click the image to play the video clip in a browser.*

Future video prediction. We show an extension of our approach to the future video prediction task: learning to predict the future video given a few observed frames. We decompose the task into two sub-tasks: 1) synthesizing future semantic segmentation masks using the observed frames, and 2) converting the synthesized segmentation masks into videos. In practice, after extracting the segmentation masks from the observed frames, we train a generator to predict future semantic masks. We then use the proposed video-to-video synthesis approach to convert the predicted segmentation masks to a future video.

We conduct both quantitative and qualitative evaluations with comparisons to two start-of-the-art approaches: PredNet [45] and MCNet [68]. We follow the prior work [41, 70] and report the human preference score. We also include the FID scores. As shown in Table 3, our model produces smaller FIDs, and the human subjects favor our resulting videos. In Figure 7, we visualize the future video synthesis results. While the image quality of the results from the competing algorithms degrades significantly over time, ours remains consistent.

5 Discussion

We present a general video-to-video synthesis framework based on conditional GANs. Through carefully-designed generators and discriminators as well as a spatio-temporal adversarial objective, we can synthesize high-resolution, photorealistic, and temporally consistent videos. Extensive experiments demonstrate that our results are significantly better than the results by state-of-the-art methods. Our method also compares favorably against the competing video prediction methods.

Although our approach outperforms previous methods, our model still fails in a couple of situations. For example, our model struggles in synthesizing turning cars due to insufficient information in label maps. This could be potentially addressed by adding additional 3D cues, such as depth maps. Furthermore, our model still can not guarantee that an object has a consistent appearance across the whole video. Occasionally, a car may change its color gradually. This issue might be alleviated if object tracking information is used to enforce that the same object shares the same appearance throughout the entire video. Finally, when we perform semantic manipulations such as turning trees into buildings, visible artifacts occasionally appear as building and trees have different label shapes. This might be resolved if we train our model with coarser semantic labels, as the trained model would be less sensitive to label shapes.

Acknowledgements We thank Karan Sapra, Fitsum Reda, and Matthieu Le for generating the segmentation maps for us. We also thank Lisa Rhee and Miss Ketsuki for allowing us to use their dance videos for training. We thank William S. Peebles for proofreading the paper.

References

[1] K. Aberman, M. Shi, J. Liao, D. Lischinski, B. Chen, and D. Cohen-Or. Deep video-based performance cloning. *arXiv preprint arXiv:1808.06847*, 2018.

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[3] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: robust video object cutout using localized classifiers. *ACM Transactions on Graphics (TOG)*, 28(3):70, 2009.

[4] G. Balakrishnan, A. Zhao, A. V. Dalca, F. Durand, and J. Guttag. Synthesizing images of humans in unseen poses. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] D. Bitouk, N. Kumar, S. Dhillon, P. Belhumeur, and S. K. Nayar. Face swapping: automatically replacing faces in photographs. In *ACM SIGGRAPH*, 2008.

[6] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[7] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros. Everybody dance now. In *European Conference on Computer Vision (ECCV) Workshop*, 2018.

[10] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua. Coherent online video style transfer. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[11] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[12] T. Chen, J.-Y. Zhu, A. Shamir, and S.-M. Hu. Motion-aware gradient domain video composition. *IEEE Trans. Image Processing*, 22(7):2532–2544, 2013.

[13] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a Laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[15] E. L. Denton and V. Birodkar. Unsupervised learning of disentangled representations from video. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[16] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[17] P. Esser, E. Sutter, and B. Ommer. A variational u-net for conditional appearance and shape generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[19] A. Ghosh, V. Kulharia, V. Namboodiri, P. H. Torr, and P. K. Dokania. Multi-agent diverse generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[21] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein GANs. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[22] A. Gupta, J. Johnson, A. Alahi, and L. Fei-Fei. Characterizing and improving stability in neural style transfer. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[23] R. A. Güler, N. Neverova, and I. Kokkinos. Densepose: Dense human pose estimation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[24] D. Ha and J. Schmidhuber. World models. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[25] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[27] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[28] H. Huang, H. Wang, W. Luo, L. Ma, W. Jiang, X. Zhu, Z. Li, and W. Liu. Real-time neural style transfer for videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[29] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang. The ApolloScape dataset for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[30] X. Huang, Y. Li, O. Poursaeed, J. E. Hopcroft, and S. J. Belongie. Stacked generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[32] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[33] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[34] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, 2016.

[35] J. T. Kajiya. The rendering equation. In *ACM SIGGRAPH*, 1986.

[36] N. Kalchbrenner, A. v. d. Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016.

[37] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018.

[38] D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 2009.

[39] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[40] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning (ICML)*, 2016.

[41] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

[42] X. Liang, L. Lee, W. Dai, and E. P. Xing. Dual motion GAN for future-flow embedded video prediction. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[43] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[44] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[45] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[46] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. *International Joint Conference on Artificial Intelligence (IJCAI)*, 1981.

[47] L. Ma, X. Jia, Q. Sun, B. Schiele, T. Tuytelaars, and L. Van Gool. Pose guided person image generation. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[48] L. Ma, Q. Sun, S. Georgoulis, L. Van Gool, B. Schiele, and M. Fritz. Disentangled person image generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[49] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[50] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *International Conference on Learning Representations (ICLR)*, 2016.

[51] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[52] T. Miyato and M. Koyama. cGANs with projection discriminator. In *International Conference on Learning Representations (ICLR)*, 2018.

[53] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier GANs. In *International Conference on Machine Learning (ICML)*, 2017.

[54] K. Ohnishi, S. Yamamoto, Y. Ushiku, and T. Harada. Hierarchical video generation from orthogonal information: Optical flow and texture. In *AAAI*, 2018.

[55] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2015.

[56] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning (ICML)*, 2016.

[57] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*, 2018.

[58] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In *German Conference on Pattern Recognition*, 2016.

[59] M. Saito, E. Matsumoto, and S. Saito. Temporal generative adversarial nets with singular value clipping. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[60] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. *ACM Transactions on Graphics (TOG)*, 2000.

[61] E. Shechtman, Y. Caspi, and M. Irani. Space-time super-resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(4):531–545, 2005.

[62] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[63] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[64] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[65] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning (ICML)*, 2015.

[66] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. In *International Conference on Learning Representations (ICLR)*, 2017.

[67] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. MoCoGAN: Decomposing motion and content for video generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[68] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing motion and content for natural video sequence prediction. In *International Conference on Learning Representations (ICLR)*, 2017.

[69] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[70] C. Vondrick and A. Torralba. Generating the future with adversarial transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[71] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision (ECCV)*, 2016.

[72] J. Walker, K. Marino, A. Gupta, and M. Hebert. The pose knows: Video forecasting by generating pose futures. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[73] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[74] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[75] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(3), 2007.

[76] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[77] T. Xue, J. Wu, K. Bouman, and B. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[78] C. Yang, Z. Wang, X. Zhu, C. Huang, J. Shi, and D. Lin. Pose guided human video generation. In *European Conference on Computer Vision (ECCV)*, 2018.

[79] S. Yang, T. Ambert, Z. Pan, K. Wang, L. Yu, T. Berg, and M. C. Lin. Detailed garment recovery from a single-view image. *arXiv preprint arXiv:1608.01250*, 2016.

[80] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[81] Z.-H. Zheng, H.-T. Zhang, F.-L. Zhang, and T.-J. Mu. Image-based clothes changing system. *Computational Visual Media*, 3(4):337–347, 2017.

[82] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[83] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

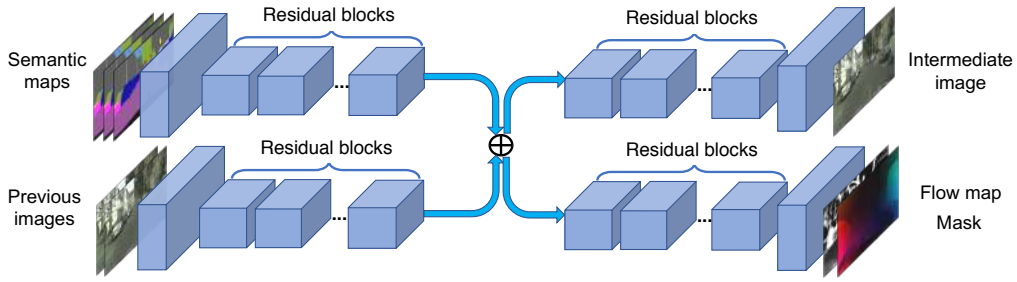


Figure 8: The network architecture (G_1) for low-res videos. Our network takes in a number of semantic label maps and previously generated images, and outputs the intermediate frame as well as the flow map and the mask.

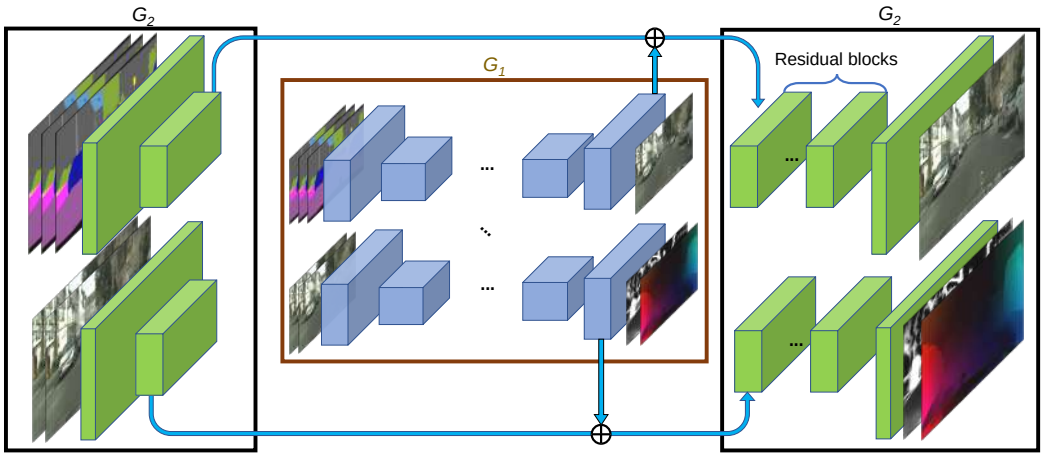


Figure 9: The network architecture (G_2) for higher resolution videos. The label maps and previous frames are downsampled and fed into the low-res network G_1. Then, the features from the high-res network and the last layer of the low-res network are summed and fed into another series of residual blocks to output the final images.

A Network Architecture

A.1 Generators

Our network adopts a coarse-to-fine architecture. For the lowest resolution, the network takes in a number of semantic label maps $\mathbf{s}_{t-L}^{t}$ and previously generated frames $\tilde{\mathbf{x}}_{t-L}^{t-1}$ as input. The label maps are concatenated together and undergo several residual blocks to form intermediate high-level features. We apply the same processing for the previously generated images. Then, these two intermediate layers are added and fed into two separate residual networks to output the hallucinated image $\tilde{\mathbf{h}}_t$ as well as the flow map $\tilde{\mathbf{w}}_t$ and the mask $\tilde{\mathbf{m}}_t$ (Figure 8).

Next, to build from low-res results to higher-res results, we use another network G_2 on top of the low-res network G_1 (Figure 9). In particular, we first downsample the inputs and fed them into G_1. Then, we extract features from the last feature layer of G_1 and add them to the intermediate feature layer of G_2. These summed features are then fed into another series of residual blocks to output the higher resolution images.

A.2 Discriminators

For our image discriminator D_I, we adopt the multi-scale PatchGAN architecture [33, 73]. We also design a temporally multi-scale video discriminator D_V by downsampling the frame rates of the real/generated videos. In the finest scale, the discriminator takes K consecutive frames in the original sequence as input. In the next scale, we subsample the video by a factor of K (i.e., skipping every

$K-1$ intermediate frames), and the discriminator takes consecutive K frames in this new sequence as input. We do this for up to three scales in our implementation and find that this helps us ensure both short-term and long-term consistency. Note that D_V is also multi-scale in the spatial domain as D_I.

A.3 Feature matching loss

In our learning objective function, we also add VGG feature matching loss and discriminator feature matching loss to improve the training stability. For VGG feature matching loss, we use the VGG network [64] as a feature extractor and minimize L1 losses between the extracted features from the real and the generated images. In particular, we add $\sum_i \frac{1}{P_i}[||\psi^{(i)}(\mathbf{x}) - \psi^{(i)}(G(\mathbf{s}))||_1]$ to our objective, where $\psi^{(i)}$ denotes the i-th layer with P_i elements of the VGG network. Similarly, we adopt the discriminator feature matching loss, to match the statistics of features extracted by the GAN discriminators. We use both the image discriminator D_I and the video discriminator D_V.

B Evaluation for the Apolloscape Dataset

We provide both the FID and the human preference score on the Apolloscape dataset. For both metrics, our method outperforms the other baselines.

Table 4: Comparison between competing video-to-video synthesis approaches on Apolloscape.

Inception Net. of FID	I3D	ResNeXt
`pix2pixHD`	2.33	0.128
`COVST`	2.36	0.128
`vid2vid` (ours)	**2.24**	**0.125**

Human Preference Score	
`vid2vid` (ours) / `pix2pixHD`	**0.61** / 0.39
`vid2vid` (ours) / `COVST`	**0.59** / 0.41

Everybody Dance Now

Caroline Chan* Shiry Ginosar Tinghui Zhou† Alexei A. Efros

UC Berkeley

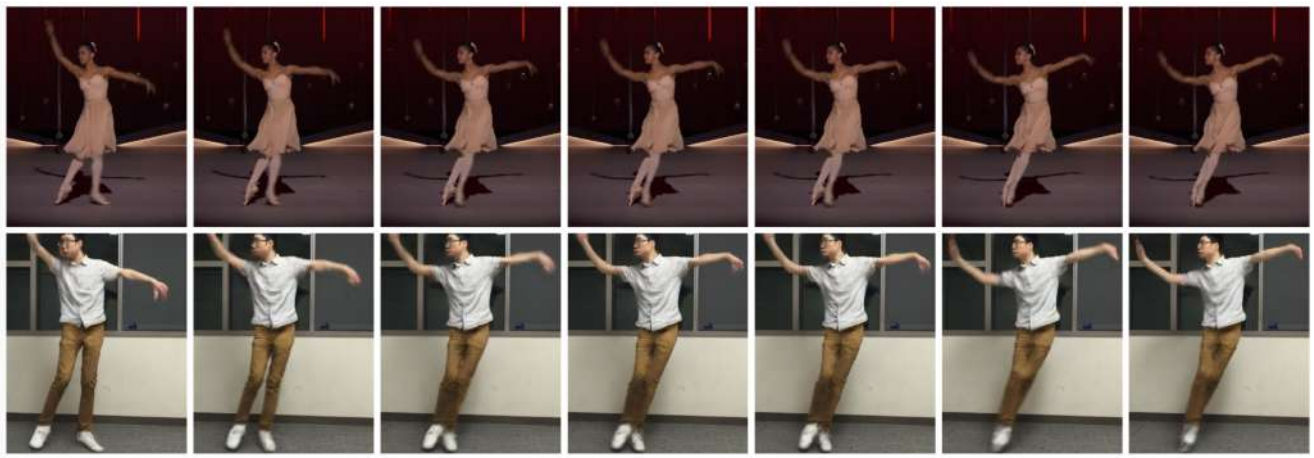


Figure 1: **"Do as I Do" motion transfer:** given a YouTube clip of a ballerina (top), and a video of a graduate student performing various motions, our method transfers the ballerina's performance onto the student (bottom). Video: `https://youtu.be/mSaIrz8lM1U`

Abstract

This paper presents a simple method for "do as I do" motion transfer: given a source video of a person dancing, we can transfer that performance to a novel (amateur) target after only a few minutes of the target subject performing standard moves. We approach this problem as video-to-video translation using pose as an intermediate representation. To transfer the motion, we extract poses from the source subject and apply the learned pose-to-appearance mapping to generate the target subject. We predict two consecutive frames for temporally coherent video results and introduce a separate pipeline for realistic face synthesis. Although our method is quite simple, it produces surprisingly compelling results (see video). This motivates us to also provide a forensics tool for reliable synthetic content detection, which is able to distinguish videos synthesized by our system from real data. In addition, we release a first-of-its-kind open-source dataset of videos that can be legally used for training and motion transfer.

1. Introduction

Consider the two video sequences on Figure 1. The top row is the input – it is a YouTube clip of a ballerina (the *source* subject) performing a sequence of motions. The bottom row is the output of our algorithm. It corresponds to frames of a different person (the *target* subject) apparently performing the same motions. The twist is that the target person never performed the same exact sequence of motions as the source, and, indeed, knows nothing about ballet. He was instead filmed performing a set of standard moves, without specific reference to the precise actions of the source. And, as is obvious from the figure, the source and the target are of different genders, have different builds, and wear different clothing.

In this work, we propose a simple but surprisingly effective approach for "Do as I Do" video retargeting – automatically transferring the motion from a source to a target subject. Given two videos – one of a *target* person whose appearance we wish to synthesize, and the other of a *source* subject whose motion we wish to impose onto our target person – we transfer motion between these subjects by learning a simple video-to-video translation. With our framework, we create a variety of videos, enabling un-

*C. Chan is currently a graduate student at MIT CSAIL.

†T. Zhou is currently affiliated with Humen, Inc.

trained amateurs to spin and twirl like ballerinas, perform martial arts kicks, or dance as vibrantly as pop stars.

To transfer motion between two video subjects in a frame-by-frame manner, we must learn a mapping between images of the two individuals. Our goal is, therefore, to discover an image-to-image translation [16] between the source and target sets. However, we do not have corresponding pairs of images of the two subjects performing the same motions to supervise learning this translation. Even if both subjects perform the same routine, it is still unlikely to have an exact frame to frame pose correspondence due to body shape and motion style unique to each subject.

We observe that keypoint-based pose preserves motion signatures over time while abstracting away as much subject identity as possible and can serve as an intermediate representation between any two subjects. We therefore use pose stick figures obtained from off-the-shelf human pose detectors, such as OpenPose [6, 34, 43], as an intermediate representation for frame-to-frame transfer, as shown in Figure 2. We then learn an image-to-image translation model between pose stick figures and images of our target person. To transfer motion from source to target, we input the pose stick figures from the source into the trained model to obtain images of the target subject in the same pose as the source.

The central contribution of our work is a surprisingly simple method for generating compelling results on human motion transfer. We demonstrate complex motion transfer from realistic in-the-wild input videos and synthesize high-quality and detailed outputs (see Section 4.3 and our video for examples). Motivated by the high quality of our results, we introduce an application for detecting if a video is real or synthesized by our method. We strongly believe that it is important for work in image synthesis to explicitly address the issue of fake detection (Section 5).

Furthermore, we release a two-part dataset: First, five long single-dancer videos which we filmed ourselves that can be used to train and evaluate our model, and second, a large collection of short YouTube videos that can be used for transfer and fake detection. We specifically designate the single-dancer data to be high-resolution open-source data for training motion transfer and video generation methods. The subjects whose data we release have all consented to allowing the data to be used for research purposes. For more details, see our project website `https://carolineec.github.io/everybody_dance_now`.

2. Related Work

Over the last two decades there has been extensive work dedicated to motion transfer. Early methods focused on creating new content by manipulating existing video footage [5, 12, 31]. For example, Video Rewrite [5] creates videos of a subject saying a phrase they did not originally utter by finding frames where the mouth position matches the desired speech. Efros et al. [12] use optical flow as a descriptor to match different subjects performing similar actions allowing "Do as I do" and "Do as I say" retargeting. Classic computer graphics approaches to motion transfer attempt to perform this in 3D. Ever since the retargeting problem was proposed between animated characters [14], solutions have included the use of inverse kinematic solvers [23] and retargeting between significantly different 3D skeletons [15]. Our approach is similarly designed for in-the-wild video subjects, although we learn to synthesize novel motions rather than manipulating existing frames and we use 2D representations.

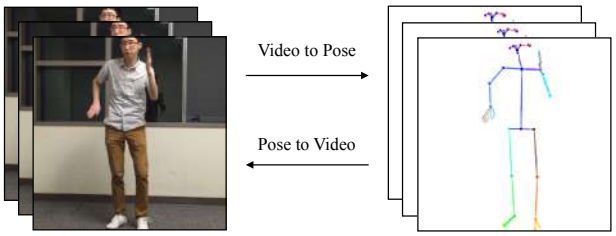


Figure 2: Our method creates correspondences by detecting poses in video frames (Video to Pose) and then learns to generate images of the target subject from the estimated pose (Pose to Video).

Several approaches rely on calibrated multi-camera setups to 'scan' a target actor and manipulate their motions in a new video through a fitted 3D model of the target. To obtain 3D information, Cheung et al. [9] propose an elaborate multi-view system to calibrate a personalized kinematic model, obtain 3D joint estimations, and render images of a human subject performing new motions. Xu et al. [45] use multi-view captures of a target subject performing simple motions to create a database of images and transfer motion through a fitted 3D skeleton and corresponding surface mesh for the target. Work by Casas et al. use 4D Video Textures [7] to compactly store a layered texture representation of a scanned target person and use their temporally coherent mesh and data representation to render video of the target subject performing novel motions. In contrast, our approach explores motion transfer between 2D video subjects and avoid data calibration and lifting into 3D space.

Similarly to our method, recent works have applied deep learning for reanimation in different applications and rely on more detailed input representations. Given synthetic renderings, an interior face model, and a gaze map as input, Kim et al. [19] transfer head position and facial expressions between human subjects and render their results in detailed portrait videos. Our problem is analogous to this work except we retarget full body motion, and the inputs to our model as 2D pose stick figures as opposed to more detailed 3D representations. Similarly, Martin-Brualla et al. [29] apply neural re-rendering to enhance rendering of human motion capture for VR/AR purposes. The primary

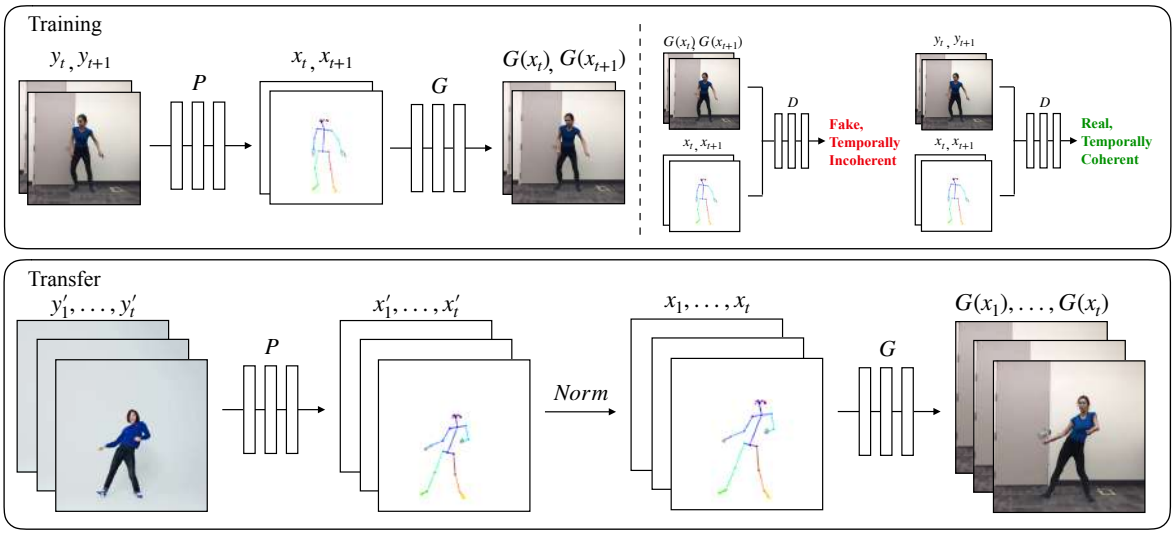


Figure 3: (Top) **Training**: Our model uses a pose detector P to create pose stick figures from video frames of the target subject. We learn the mapping G alongside an adversarial discriminator D which attempts to distinguish between the "real" correspondences $(x_t, x_{t+1}), (y_t, y_{t+1})$ and the "fake" sequence $(x_t, x_{t+1}), (G(x_t), G(x_{t+1}))$. (Bottom) **Transfer**: We use a pose detector P to obtain pose joints for the source person that are transformed by our normalization process $Norm$ into joints for the target person for which pose stick figures are created. Then we apply the trained mapping G.

focus of this work is to render realistic humans in real time and similarly uses a deep network to synthesize their final result, but unlike our work does not address motion transfer between subjects. Villegas et al. [37] focus on retargeting motion between rigged skeletons and demonstrate reanimation in 3D characters without supervised data. Similarly, we learn to retarget motion using a skeleton-like intermediate representation, however we transfer full body motion between human subjects who are not rigged to the skeleton unlike animated characters.

Recent methods focus on disentangling motion from appearance and synthesizing videos with novel motion [36, 2]. MoCoGAN [36] employs unsupervised adversarial training to learn this separation and generates videos of subjects performing novel motions or facial expressions. This theme is continued in Dynamics Transfer GAN [2] which transfers facial expressions from a source subject in a video onto a target person given in a static image. Similarly, we apply our representation of motion to different target subjects to generate new motions. However, in contrast to these methods we specialize on synthesizing detailed dance videos.

Modern approaches have shown success in generating detailed single images of human subjects in new poses [3, 10, 11, 18, 22, 27, 28, 33, 38, 13, 46]. Works including Ma et al. [27, 28] and Siarohin et al. [33] have introduced novel architectures and losses for this purpose. Furthermore, [39, 38] have shown pose is an effective supervisory signal for future prediction and video generation. However these works are not designed specifically for motion transfer. Rather than generating possible views of a previously unseen person from a single input image, we are interested in learning the style of a single, known person from large amounts of personalized video data and synthesizing them dancing in a detailed high-resolution video.

Concurrent with our work, [1, 4, 24, 40] learn mappings between videos and demonstrate motion transfer between faces and from poses to body. Wang et al. [40] achieves results of similar quality to ours with a more complex method and significantly more computational resources.

Our work is made possible by recent rapid advances along two separate directions: robust pose estimation, and realistic image-to-image translation. Modern pose detection systems including OpenPose [6, 34, 43] and DensePose [32] allow for surprisingly reliable and fast pose extraction in a variety of scenarios. At the same time, the recent emergence of image-to-image translation models, pix2pix [16], CoGAN [26], UNIT [25], CycleGAN [48], DiscoGAN [20], Cascaded Refinement Networks [8], and pix2pixHD [41], have enabled high-quality single-image generation. We build upon these two building blocks by using pose detection as an intermediate representation and extending upon single-image generation to synthesize temporally-coherent, surprisingly realistic videos.

3. Method

Given a video of a source person and another of a target person, our goal is to generate a new video of the target enacting the same motions as the source. To accomplish this task, we divide our pipeline into three stages – pose detection, global pose normalization, and mapping from normalized pose stick figures to the target subject. See Figure 3 for

an overview of our pipeline. In the pose detection stage we use a pre-trained state-of-the-art pose detector to create pose stick figures given frames from the source video. The global pose normalization stage accounts for differences between the source and target body shapes and locations within the frame. Finally, we design a system to learn the mapping from the pose stick figures to images of the target person using adversarial training. Next we describe each stage of our system.

3.1. Pose Encoding and Normalization

Encoding body poses To encode the body pose of a subject image, we use a pre-trained pose detector P (OpenPose [6, 34, 43]) which accurately estimates 2D x, y joint coordinates. We then create a colored pose stick figure by plotting the keypoints and drawing lines between connected joints as shown in Figure 2.

Global pose normalization In different videos, subjects may have different limb proportions or stand closer or farther to the camera than one another. Therefore when retargeting motion between two subjects, it may be necessary to transform the pose keypoints of the source person so that they appear in accordance with the target person's body shape and location as in the **Transfer** section of Figure 3. We find this transformation by analyzing the heights and ankle positions for the poses of each subject and use a linear mapping between the closest and farthest ankle positions in both videos. After gathering these positions, we calculate the scale and translation for each frame based on its corresponding pose detection. Details of this process are described in Section 8.5.

3.2. Pose to Video Translation

Our video synthesis method is based off of an adversarial single frame generation process presented by Wang et al. [41]. In the original conditional GAN setup, the generator network G engages in a minimax game against multiscale discriminator $D = (D_1, D_2, D_3)$. The generator must synthesize images in order to fool the discriminator which must discern between "real" (ground truth) images and "fake" images produced by the generator. The two networks are trained simultaneously and drive each other to improve - Gr learns to synthesize more detailed images to deceive D which in turn learns differences between generated outputs and ground truth data. For our purposes, G synthesizes images of a person given a pose stick figure.

Such single-frame image-to-image translation methods are not suitable for video synthesis as they produce temporal artifacts and cannot generate the fine details important in perceiving humans in motion. We therefore add a learned model of temporal coherence as well as a module for high resolution face generation.

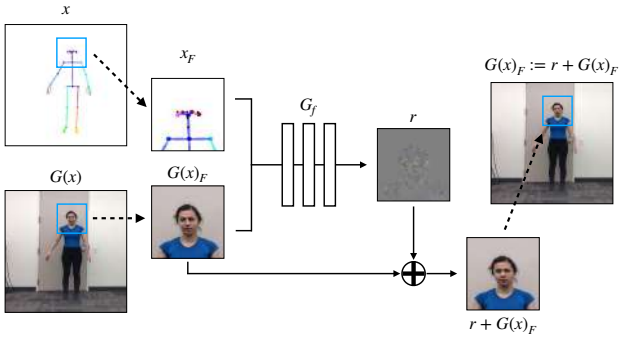


Figure 4: Face GAN setup. Residual is predicted by generator G_f and added to the original face prediction from the main generator.

Temporal smoothing To create video sequences, we modify the single image generation setup to enforce temporal coherence between adjacent frames as shown in Figure 3 (top right). Instead of generating individual frames, we predict two consecutive frames where the first output $G(x_{t-1})$ is conditioned on its corresponding pose stick figure x_{t-1} and a zero image z (a placeholder since there is no previously generated frame at time $t-2$). The second output $G(x_t)$ is conditioned on its corresponding pose stick figure x_t and the first output $G(x_{t-1})$. Consequently, the discriminator is now tasked with determining both the difference in realism and temporal coherence between the "fake" sequence $(x_{t-1}, x_t, G(x_{t-1}), G(x_t))$ and "real" sequence $(x_{t-1}, x_t, y_{t-1}, y_t)$. The temporal smoothing changes are now reflected in the updated GAN objective

$$\mathcal{L}_{\text{smooth}}(G, D) = \mathbb{E}_{(x,y)}[\log D(x_t, x_{t+1}, y_t, y_{t+1})] + \mathbb{E}_x[\log(1 - D(x_t, x_{t+1}, G(x_t), G(x_{t+1})))] \quad (1)$$

Face GAN We add a specialized GAN setup to add more detail and realism to the face region as shown in Figure 4. After generating the full image of the scene with the main generator G, we input a smaller section of the image centered around the face (i.e. 128×128 patch centered around the nose keypoint), $G(x)_F$, and the input pose stick figure sectioned in the same fashion, x_F, to another generator G_f which outputs a residual $r = G_f(x_F, G(x)_F)$. The final synthesized face region is the addition of the residual with the face region of the main generator $r + G(x)_F$. A discriminator D_f then attempts to discern the "real" face pairs (x_F, y_F) from the "fake" face pairs $(x_F, r + G(x)_F)$, similarly to the original pix2pix [16] objective:

$$\mathcal{L}_{\text{face}}(G_f, D_f) = \mathbb{E}_{(x_F, y_F)}[\log D_f(x_F, y_F)] + \mathbb{E}_{x_F}[\log\left(1 - D_f(x_F, G(x)_F + r)\right)]. \quad (2)$$

Here x_F is the face region of the original pose stick figure x and y_F is the face region of ground truth target person image y. Similarly to the full image, we add a perceptual reconstruction loss on comparing the final face $r + G(x)_F$ to the ground truth target person's face y_F.

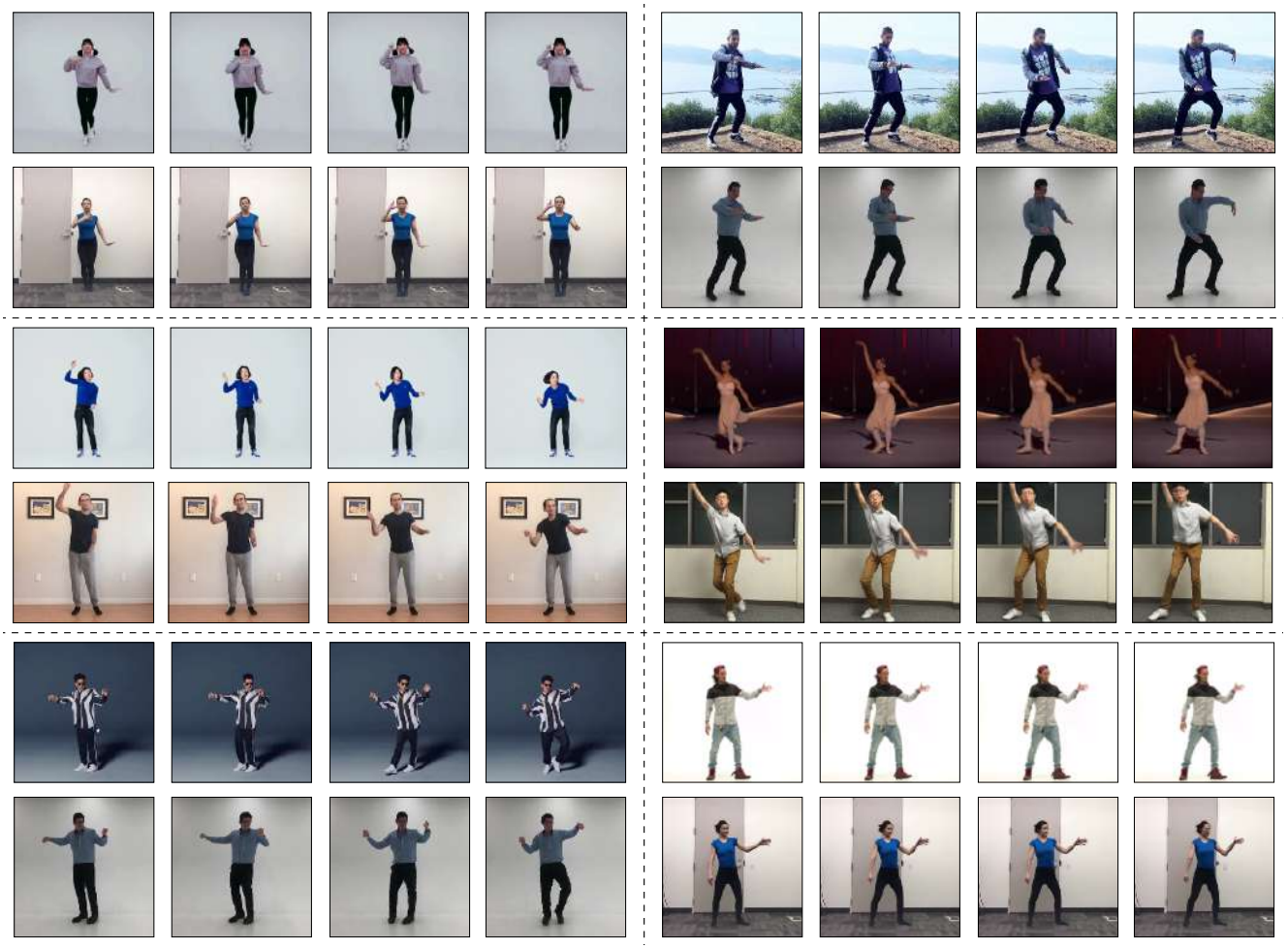


Figure 5: Transfer results. In each section we show four consecutive frames. The top row shows the source subject and the bottom row shows the synthesized outputs of the target person.

3.3. Full Objective

We employ training in stages where the full image GAN is optimized separately from the specialized face GAN. First we train the main generator and discriminator (G, D) during which the full objective is -

$$\min_{G}((\max_{D_i}\sum_{k_i}\mathcal{L}_{\text{smooth}}(G, D_k)) + \lambda_{FM}\sum_{k_i}\mathcal{L}_{\text{FM}}(G, D_k) + \lambda_P(\mathcal{L}_P(G(x_{t-1}), y_{t-1}) + \mathcal{L}_P(G(x_t), y_t))) \quad (3)$$

Where $i = 1, 2, 3$. Here, $\mathcal{L}_{\text{GAN}}(G, D)$ is the single image adversarial loss presented in the original pix2pix paper [16]:

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{(x,y)}[\log D(x, y)] + \mathbb{E}_x[\log(1 - D(x, G(x))] \quad (4)$$

$\mathcal{L}_{\text{FM}}(G, D)$ is the discriminator feature-matching loss presented in pix2pixHD, and $\mathcal{L}_P(G(x), y)$ is the perceptual reconstruction loss [17] which compares pretrained VGGNet [35] features at different layers of the network (fully specified in the Section 8.2).

After this stage, the full image GAN weights are frozen and we optimize the face GAN with objective

$$\min_{G_f}\left(\left(\max_{D_f}\mathcal{L}_{\text{face}}(G_f, D_f)\right) + \lambda_P\mathcal{L}_P(r + G(x)_F, y_F)\right) \quad (5)$$

where $\mathcal{L}_{\text{FM}}(G, D)$ is the discriminator feature-matching loss presented in pix2pixHD, and $\mathcal{L}_P$ is a perceptual reconstruction loss [17] which compares pretrained VGGNet [35] features at different layers of the network. For training details see Section 8.2.

4. Experiments

We compare our performance to baseline methods on multiple target subjects and source motions.

4.1. Setup

We collect two types of data long, open-source, single-dancer *target* videos which we film ourselves to train our

model on and make publicly available, and in-the-wild *source* videos collected online for motion transfer. The filming set-up for target videos and collection method for source videos are detailed in Section 8.3.

Baseline methods 1) **Nearest Neighbors**. For each source video frame, we retrieve the closest match in the training target sequence using the following pose distance metric: For two poses p, p' each with n joints $p_1, ..., p_n$ and $p'_1, ..., p'_n$, we define the distance between them as the normalized sum of the L2 distances between the corresponding joints $p_k = (x_k, y_k)$ and $p'_k = (x'_k, y'_k)$:

$$d(p, p') = \frac{1}{n} \sum_{k=1}^{n} \|p_k - p'_k\|_2 \tag{6}$$

The adjacent target matches frames are then concatenated into a frame-by-frame nearest neighbors sequence.
2) **Balakrishnan *et al*. (PoseWarp)** [3] generate images of a given target subject in a new pose. While, unlike ours, this method is designed for single image synthesis, we use it to synthesize a video frame-by-frame for comparison.

Ablation conditions 1) **Frame-by-frame synthesis** (FBF). In this condition we ablate our temporal smoothing setup and apply pix2pixHD [41] on a per-frame basis. 2) **Temporal smoothing** (FBF+TS). In this condition we ablate the Face GAN module to study the difference it makes on the final result. 3) **Our model** (FBF+TS+FG). uses both temporal smoothing and a Face GAN.

Evaluation metrics We use perceptual studies on Mechanical Turk for evaluating the video results of our final method in comparison to ablated conditions and baselines. For the ablation study, we further measure the quality of each synthesized frame using two metrics: 1) **SSIM**. Structural Similarity [42] and 2) **LPIPS** Learned Perceptual Image Patch Similarity [47]. We examined the pose distance seen in Equation 6 to measure the similarity between input and synthesized pose. However, we found this distance to be not very informative due to noisy detections.

4.2. Quantitative Evaluation

We quantitatively compare our approach against the baselines, and then against ablated versions of our method.

4.2.1 Comparison to Baselines

We compare our method to baselines on the same transfer task for all subjects for which we filmed longer videos. From a single out-of-sample source video, we synthesize a transfer video for every baseline-subject pair. We then crop the same 10-second snippets of video for each baseline and subject pair and use these for our perceptual studies.

Method	1	2	3	4	5	Total
NN	95.9%	96.4%	94.6%	95.8%	94.7%	95.1%
PoseWarp [3]	83.1%	69.9%	88.7%	84.6%	74.4%	83.3%

Table 1: Comparison to baselines using perceptual studies for subjects 1 through 5 and in total average. We report the percentage of time participants chose **our** method as more realistic than the baseline.

Method	1	2	3	4	5	Total
NN	85%	93%	94%	90%	91%	91.2%
PoseWarp [3]	77.5%	70%	80%	90%	78.7%	79.1%

Table 2: Comparison of our method without Face GAN (FBF+TS variant) to baselines for subjects 1 through 5 and in total average. We report the percentage of time participants chose the FBF+TS ablation as more realistic than the baseline.

Participants on MTurk watched a series of video pairs. In each pair, one video was synthesized using our method; the other by a baseline. They were then asked to pick the more realistic one. Videos of resolution 144×256 (as this is the highest resolution that PoseWarp baseline can produce) were shown, and after each pair, participants were given unlimited time to respond. Each task consisted of 18 pairs of videos and was performed by 100 distinct participants. Table 1 displays the results of this study and shows that participants indicated our method is more realistic 95.1% and 83.3% of the time on average in comparison to the Nearest Neighbors and PoseWarp [3] baselines respectively.

We include an additional perceptual study to verify our method is not preferred over the others simply due to more emphasis on face synthesis. We compare the FBF+TS variant (without the Face GAN module) to both baselines in Table 2. We find that the FBF+TS ablation is consistently preferred, albeit slightly less than our full model, over the Nearest Neighbors and PoseWarp baselines 91.2% and 79.1% of the time on average respectively.

4.2.2 Ablation Study

We perform an ablation study on held-out test data of the target subject (the source and target are the same) since we do not have paired same-pose frames across subjects.

As shown in Table 3a(bottom), both SSIM and LPIPS scores are similar for all model variations on the body regions. Scores on full images are even more similar, as the ablated models have no difficulty generating the static background. However, Table 3a(top) demonstrates the effectiveness of our face residual generator by showing the improvement of our full model over the the FBF+TS condition.

As these comparisons are in a frame-by-frame fashion they do not emphasize the usefulness of our temporal smoothing setup. The effect of this module can be seen in the qualitative video results and in the perceptual studies

Region	Metric	FBF	FBF+TS	FBF+TS+FG
Face	SSIM	0.784	0.811	**0.816**
	LPIPS	0.045	0.039	**0.036**
Body	SSIM	0.828	**0.838**	**0.838**
	LPIPS	0.057	0.051	**0.050**

(a) Metric comparison for synthesized face (top) and full-body (bottom) regions. Metrics are averaged over the 5 subjects. For SSIM higher is better. For LPIPS lower is better.

Condition	1	2	3	4	5	Total
FBF	54.1%	69.7%	62.4%	53.8%	60.0%	58.8%
FBF+TS	59.6%	56.4%	50.3%	53.0%	53.1%	53.9%

(b) Perceptual study results for subjects 1 through 5 and in total average. We report the percentage of time participants chose **our** method as more realistic than the ablated conditions.

Table 3: Ablation studies. We compare frame-by-frame synthesis (FBF), adding temporal smoothing (FBF+TS) and our final model with temporal smoothing and Face GAN modules (FBF+TS+FG).

Condition	1	2	3	4	5	Total
Prefer FBF+TS	60.5%	62%	57.5%	50%	62.5%	58.5%

Table 4: Comparison of our method without Face GAN (FBF+TS) to the FBF ablation for subjects 1 through 5 and in total average. We report the percentage of time participants chose the FBF+TS ablation over the FBF ablation.

results in Table 3b. Here we see that our method is preferred 58.8% and 53.3% of the time over frame-by-frame synthesis and the No Face GAN (FBF+TS) setup respectively. In general, this shows that incorporating temporal information at training time positively influences video results. Although the effect of the Face GAN can be be somewhat subtle, overall this addition benefits our results, especially in the case of subject 1 whose training video is very sharp where facial details are easily visible.

We further compare our method without the Face GAN (FBF+TS) to the frame-by-frame (FBF) ablation to verify our temporal smoothing setup alone improves result quality. Table 4 reports that the FBF+TS ablation is preferred on average over the FBF alone. Note that for subject 4 FBF produced noticeable flickering, but FBF+TS introduced texture artifacts on his loose shirt (see Figure 9).

4.3. Qualitative Results

Transfer results for multiple source and target subjects can be seen in Figure 5. The advantage of using the Face GAN module can be seen in a single frame comparison in Figure 6. As mentioned, [3] is designed for single image synthesis. Nonetheless, even for a single frame transfer, we outperform [3] as we show in Figure 7.

While the above single-image and quantitative results

Figure 6: Face image comparison on held-out data. We compare frame-by-frame synthesis (FBF), adding temporal smoothing (FBF+TS) and our full model (FBF+TS+FG).

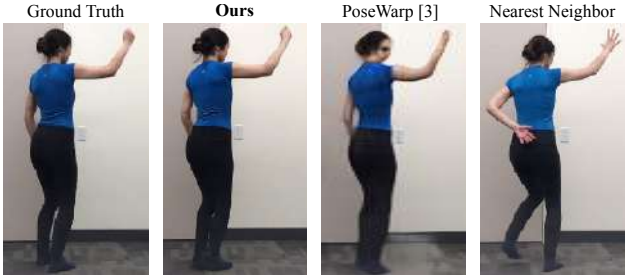


Figure 7: Comparison between our model, [3], and nearest neighbors on single-frame synthesis on held-out data.

(Section 4.2) suggest the superiority of our approach, more significant difference can be observed in our video. There we find the temporal modeling produces more frame to frame coherence than the frame-by-frame ablation, and that adding a specialized facial generator and discriminator adds considerable detail and realism.

5. Detecting Fake Videos

Recent progress on image synthesis and generative models has narrowed the gap between synthesized and real images and videos, which has raised legal and ethical questions on video authenticity (among many other social implications). Given the high quality of our results, it is important to investigate mechanisms for detecting computer-generated videos including ones generated by our model.

We train a fake-detector to identify fake videos created by our system — given a video, the fake-detector flags it as real or fake. We train the fake-detector in a parallel fashion to our synthesis process, to classify whether a sequence of 2 consecutive frames is real (from ground-truth frames) or fake (from our generation). This allows the fake-detector to exploit cues based on the fidelity of individual frames as well as consistency across time. To make a decision for the whole video in question, we multiply the decision probabilities for all consecutive frame pairs. Details of the network architecture are included in Section 8.2. For the purpose of training the fake-detector, we collect a 62-subject set of

Figure 8: Multi-subject synchronized dancing. By applying the same source motion to multiple subjects, we can create the effect of them performing synchronized dance moves.

Figure 9: Failure cases. Ground truth appearance reference (left) followed by our results (right).

Source Motion	Same subject	Mars	Copeland
Accuracy	95.68%	96.70%	97.00%

Table 5: Fake detection average accuracy for held-out target subjects. As seen in the rows, fake videos were created for each target subject using same-subject and different-subject source motions.

short 1920×1080 resolution dancing videos. This larger dataset is collected from public YouTube videos where a subject dances in front of a static camera for an average of 3 minutes. We split this set into 48 subjects for training and 14 held-out subjects for testing.

We train a separate synthesis model for each of the 48 train subjects to produce fake content for detection. By training our fake-detector on multiple fake videos depicting a large set of subjects we ensure that it generalizes to detecting fakes of different people and does not over-fit to one or two individuals. We note that since each person dancing performs a rich set of motions we require less training data than for detecting fakes in still images.

We evaluate our fake-detector on synthesized videos for 14 held-out test subjects. We use both motion taken from the same subject (where the source and target are the same person) and motion driven by a different source subject (Bruno Mars and Misty Copeland) to synthesize fake videos for each held out subject. Our results are shown in Table 5. Overall, the fake-detector successfully distinguishes real and fake sequences regardless of where the source motion is from. As expected, our fake detection accuracy is lowest for same-person motion transfer, and is highest for transfer of motion from a prima ballerina (Misty Copeland).

6. Potential Applications

One fun application of our system is to create a motion-synchronized dancing video with multiple subjects (say, for making a family reunion video). Given trained synthesis models for multiple subjects, we use the same source video to drive the motion of all target subjects — creating an effect of them performing the same dance moves in a synchronized manner. See Figure 8 and the video.

Several systems based on our prototype description were recently successfully employed commercially. One example is an augmented reality stage performance art piece where a 3D-rendered dancer appears to float next to a real dancer [30]. Another is an in-game entertainment application making NBA players dance [44].

7. Limitations and Discussion

Our relatively simple model is usually able to create arbitrarily long, good-quality videos of a target person dancing given the movements of a source dancer to follow. However, it suffers from several limitations.

We have included examples of visual artifacts in Figure 9. On the left, our model struggles with loose clothing or hair which is not conveyed well through pose. The middle columns show a missing right arm which was not detected by OpenPose. On the right we observe some texture artifacts in shirt creases. Further work could focus on improving results by combining target videos with different clothing or scene lighting, improving pose detection systems, and mitigating the artifacts caused by high frequency textures in loose/wrinkled clothing or hair.

Our pose normalization solution does not account for

different limb lengths or camera positions. These discrepancies additionally widen the gap between the motion seen in training and testing. However, our model is able to generalize to new motions fairly well from the training data. When filming a target sequence, we have no specific source motion in mind and do not require the target subject performing similar motions to any source. We instead learn a single model that generalizes to a wide range of source motion. However our model sometimes struggles to extrapolate to radically different poses. For example, artifacts can occur if the source motion contains extreme poses such as handstands if the target training data did not contain such upside-down poses. Future work could focus on the training data, i.e. what poses and how many are needed to learn a effective model. This area relates to work on understanding which training examples are most influential [21].

Acknowledgements We thank Andrew Owens for the catchy title. This work was supported, in part, by NSF grant IIS-1633310 and research gifts from Adobe, eBay, and Google.

References

[1] Kfir Aberman, Mingyi Shi, Jing Liao, D Liscbinski, Baoquan Chen, and Daniel Cohen-Or. Deep video-based performance cloning. In *Computer Graphics Forum*, volume 38, pages 219–233. Wiley Online Library, 2019.

[2] Wissam J Baddar, Geonmo Gu, Sangmin Lee, and Yong Man Ro. Dynamics transfer gan: Generating video by transferring arbitrary temporal dynamics from a source video to a single target image. *arXiv preprint arXiv:1712.03534*, 2017.

[3] Guha Balakrishnan, Amy Zhao, Adrian V Dalca, Fredo Durand, and John Guttag. Synthesizing images of humans in unseen poses. In *CVPR*, 2018.

[4] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. In *ECCV*, 2018.

[5] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video rewrite: Driving visual speech with audio. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 353–360. ACM Press/Addison-Wesley Publishing Co., 1997.

[6] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *CVPR*, 2017.

[7] Dan Casas, Marco Volino, John Collomosse, and Adrian Hilton. 4D Video Textures for Interactive Character Appearance. *Computer Graphics Forum (Proceedings of EUROGRAPHICS)*, 33(2):371–380, 2014.

[8] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, page 3, 2017.

[9] German KM Cheung, Simon Baker, Jessica Hodgins, and Takeo Kanade. Markerless human motion transfer. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pages 373–378. IEEE, 2004.

[10] Rodrigo de Bem, Arnab Ghosh, Thalaiyasingam Ajanthan, Ondrej Miksik, N Siddharth, and Philip Torr. A semi-supervised deep generative model for human body analysis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[11] Rodrigo De Bem, Arnab Ghosh, Adnane Boukhayma, Thalaiyasingam Ajanthan, N Siddharth, and Philip Torr. A conditional deep generative model of people in natural images. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1449–1458. IEEE, 2019.

[12] Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *IEEE International Conference on Computer Vision*, pages 726–733, Nice, France, 2003.

[13] Patrick Esser, Ekaterina Sutter, and Björn Ommer. A variational u-net for conditional appearance and shape generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8857–8866, 2018.

[14] Michael Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42. ACM, 1998.

[15] Chris Hecker, Bernd Raabe, Ryan W Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. In *ACM Transactions on Graphics (TOG)*, volume 27, page 27. ACM, 2008.

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[17] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

[18] Donggyu Joo, Doyeon Kim, and Junmo Kim. Generating a fusion image: One's identity and another's shape. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1635–1643, 2018.

[19] Hyeongwoo Kim, Pablo Carrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *ACM Transactions on Graphics (TOG)*, 37(4):163, 2018.

[20] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1857–1865. JMLR. org, 2017.

[21] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017.

[22] Christoph Lassner, Gerard Pons-Moll, and Peter V Gehler. A generative model of people in clothing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 853–862, 2017.

[23] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 39–48. ACM Press/Addison-Wesley Publishing Co., 1999.

[24] Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. Neural rendering and reenactment of human actor videos. *ACM Transactions on Graphics 2019 (TOG)*, 2019.

[25] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[26] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[27] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *Advances in Neural Information Processing Systems*, pages 406–416, 2017.

[28] Liqian Ma, Qianru Sun, Stamatios Georgoulis, Luc Van Gool, Bernt Schiele, and Mario Fritz. Disentangled person image generation. In *CVPR*, pages 99–108, 2018.

[29] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, and Sean Fanello. Lookingood: Enhancing performance capture with real-time neural re-rendering. *ACM Trans. Graph.*, 37(6):255:1–255:14, Dec. 2018.

[30] Kyle McDonald. Dance x Machine Learning: First Steps. `https://medium.com/@kcimc/discrete-figures-7d9e9c275c47`, 2019. [Online; accessed 21-March-2019].

[31] Greg Mori, Alex Berg, Alexei Efros, Ashley Eden, and Jitendra Malik. Video based motion synthesis by splicing and morphing. Technical Report UCB//CSD-04-1337, University of California, Berkeley, June 2004.

[32] Iasonas Kokkinos Rıza Alp Güler, Natalia Neverova. Densepose: Dense human pose estimation in the wild. *arXiv*, 2018.

[33] Aliaksandr Siarohin, Enver Sangineto, Stphane Lathuilire, and Nicu Sebe. Deformable gans for pose-based human image generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[34] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[36] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[37] Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. Neural kinematic networks for unsupervised motion retargetting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[38] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. *arXiv preprint arXiv:1704.05831*, 2017.

[39] Jacob Walker, Kenneth Marino, Abhinav Gupta, and Martial Hebert. The pose knows: Video forecasting by generating pose futures. In *International Conference on Computer Vision*, 2017.

[40] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[41] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[42] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[43] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[44] Xpire. Using AI to make NBA players dance. `https://tinyurl.com/y3bdj5p5`, 2019. [Online; accessed 21-March-2019].

[45] Feng Xu, Yebin Liu, Carsten Stoll, James Tompkin, Gaurav Bharaj, Qionghai Dai, Hans-Peter Seidel, Jan Kautz, and Christian Theobalt. Video-based characters: creating new human performances from a multi-view video database. In *ACM Transactions on Graphics (TOG)*, volume 30, page 32. ACM, 2011.

[46] Mihai Zanfir, Alin-Ionut Popa, Andrei Zanfir, and Cristian Sminchisescu. Human appearance transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5391–5399, 2018.

[47] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[48] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

8. Appendix

8.1. Video Demonstration

Our video demo can be found at `https://youtu.be/mSaIrz8lM1U` and examples from our comparison to baselines and ablation study can be found at `https://youtu.be/sQD0WVS0blg`.

8.2. Implementation Details

Our generator and discriminator architectures are modified from pix2pixHD [41] to handle the temporal set-

ting. We follow the progressive learning schedule from pix2pixHD and learn to synthesize at 512×256 at the first (global) stage, and then upsample to 1024×512 at the second (local) stage. For predicting face residuals, we use the global generator of pix2pixHD and a single 70×70 PatchGAN discriminator [16]. We set hyperparameters $\lambda_P = 5$ and $\lambda_{VGG} = 10$ during the global and local training stages respectively. For the dataset collected in Section 4.1, we trained the global stage for 5 epochs, the local stage for 30 epochs, and the face GAN for 5 epochs.

For the perceptual loss $\mathcal{L}_P$, we compare the `conv1_1`, `conv2_1`, `conv3_1`, `conv4_1`, and `conv5_1` layer outputs of the VGG-19 network.

Our generator and discriminator architectures follow that presented by Wang et al. [41]. The fake-detector architectures matches that of the discriminator with a final fully connected layer.

8.3. Dataset Collection

Our dataset of long target videos consists of footage we filmed ourselves from 8 to 17 minutes with 4 videos at 1920×1080 resolution and 1 at 1280×720. Our goal in collecting a dataset of target videos is to provide the community with open-source data for which we explicitly collect release forms in which subjects allow their data to be released to other researchers. We recruited target subjects from different sources: friends, professional dancers, reporters etc. To learn the appearance of the target subject in many poses, it is important that the target video captures a sufficient range of motion and sharp frames with minimal blur. Similarly, we used a stationary camera to ensure a static background in all frames. To ensure the quality of the frames, we filmed our target subjects for between 8 and 30 minutes of real time footage at 120 frames per second using a modern cellphone camera, and use the first 20% of the footage for training and the last 80% for testing. Since our pose representation does not encode information about clothes and hair, we instructed our target subjects not to wear loose clothing and to tie up long hair.

In contrast, source videos can be easily collected online as we only require decent pose detections on these. We therefore use in-the-wild single-dancer videos where the only restriction we enforce is a static camera position.

8.4. Comparison with vid2vid

We also compare our model with a concurrent video synthesis framework called vid2vid [40]. The excessive requirement of memory and computing power of vid2vid prohibits us from comparing with their model in the high resolution setup. Instead, we train both our model and theirs in lower resolution (512×256). Our system and vid2vid generally perform similarly and produce results of comparable quality. We provide a qualitative comparison in Figure 10.

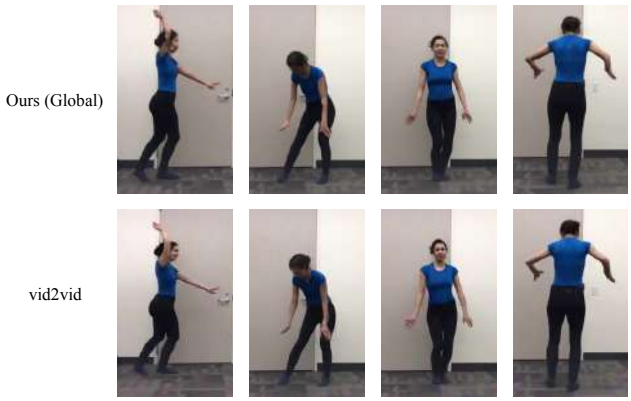


Figure 10: We compare a lower resolution version of our model without a Face GAN (top) with a lower resolution of vid2vid [41]. We find our results comparable.

8.5. Global Pose Normalization Details

In this section we describe our normalization method to match poses between the source and target. Consider a case where the source subject is significantly taller in frame than the target or is slightly elevated above the target subject's in frame position. If we directly input the unmodified poses to our system, we may generate images of the target person which are not congruent with the scene. In this example, the target person may appear large with respect to the background or surrounding objects, and may appear to be levitating since the input pose places the feet above the floor. Additionally, when generating an image from a very different pose from the in proportion and reasonably positioned poses in training, the overall quality of synthesis is expected to decline. Therefore we design a method to reasonably match the poses by finding a suitable transformation between the source and target poses. We parametrize this transformation in terms of a scale and translation factor applied to all pose keypoints for a given frame.

To find a suitable translation factor, we need to determine the position of both subjects within their respective frames. We first find the closest position s_{close} and farthest position s_{far} the source subject is away from the camera in their video. Similarly, we do the same for the target by determining t_{close} and t_{far} respectively. The goal is then to map the close and far range of the source to that of the target subject as to match the positions of both subjects, i.e. $s_{far} \mapsto t_{far}$ and $s_{close} \mapsto t_{close}$. Given a frame where the source is at position y, we then translate the source's pose vertically by:

$$translation = t_{far} + \frac{y - s_{far}}{s_{close} - s_{far}}(t_{close} - t_{far}) \quad (7)$$

In practice, we use the average of the y coordinates of the subject's ankles to determine the position within a given frame.

To reasonably scale the source poses, we determine the

heights of each subject at their closest and farthest positions in their video - denote these quantities as $h_{s_{close}}, h_{s_{far}}$ for the source and $h_{t_{close}}, h_{t_{far}}$ for the target subjects respectively. We then determine separate scales for the close position given by $c_{close} = \frac{h_{t_{close}}}{h_{s_{close}}}$ and similarly for the far position given by $c_{far} = \frac{h_{t_{far}}}{h_{s_{far}}}$. When given a frame where the source is at position y, we scale the source's pose (in both x, y directions) by:

$$scale = c_{far} + \frac{y - s_{far}}{s_{close} - s_{far}}(c_{close} - c_{far}) \quad (8)$$

We use the euclidean distance between the average ankle position and the nose keypoint of our given pose as the subject's height in a given frame.

After the translation and scale factors have been determined for a given source pose, we then add the translation to all keypoints and then apply the scale factor so that the ankle y positions remain the same (i.e. the ground is the x axis).

Given poses from a subject, we find the close position by taking the maximum y coordinate of their average ankle position over all frames.

$$s_{close} = \max\{\frac{s_{ankle1} + s_{ankle2}}{2}\}$$

The far position is found by clustering the y ankle coordinates which are less than (or spatially above) the median ankle position and about the same distance as the maximum ankle position's distance to the median ankle position. If we denote $S = \frac{s_{ankle1} + s_{ankle2}}{2}$ as the average ankle position in a given frame, then the clustering is as described by the set

$$\max\{S : ||S - s_{med}| < \alpha|s_{close} - s_{med}||\} \cap \{S < s_{med}\} \quad (9)$$

where s_{med} is the median foot position, max is the maximum ankle position, and ϵ and α are scalars. In practice we find setting $\alpha = 0.7$ generally works well, although this scalar can be finetuned on a case by case basis since it depends highly on the camera height and the subject's range of motion.

LARGE SCALE GAN TRAINING FOR HIGH FIDELITY NATURAL IMAGE SYNTHESIS

Andrew Brock[*†]
Heriot-Watt University
ajb5@hw.ac.uk

Jeff Donahue[†]
DeepMind
jeffdonahue@google.com

Karen Simonyan[†]
DeepMind
simonyan@google.com

ABSTRACT

Despite recent progress in generative image modeling, successfully generating high-resolution, diverse samples from complex datasets such as ImageNet remains an elusive goal. To this end, we train Generative Adversarial Networks at the largest scale yet attempted, and study the instabilities specific to such scale. We find that applying orthogonal regularization to the generator renders it amenable to a simple "truncation trick," allowing fine control over the trade-off between sample fidelity and variety by reducing the variance of the Generator's input. Our modifications lead to models which set the new state of the art in class-conditional image synthesis. When trained on ImageNet at 128×128 resolution, our models (BigGANs) achieve an Inception Score (IS) of 166.5 and Fréchet Inception Distance (FID) of 7.4, improving over the previous best IS of 52.52 and FID of 18.65.

1 INTRODUCTION

Figure 1: Class-conditional samples generated by our model.

The state of generative image modeling has advanced dramatically in recent years, with Generative Adversarial Networks (GANs, Goodfellow et al. (2014)) at the forefront of efforts to generate high-fidelity, diverse images with models learned directly from data. GAN training is dynamic, and sensitive to nearly every aspect of its setup (from optimization parameters to model architecture), but a torrent of research has yielded empirical and theoretical insights enabling stable training in a variety of settings. Despite this progress, the current state of the art in conditional ImageNet modeling (Zhang et al., 2018) achieves an Inception Score (Salimans et al., 2016) of 52.5, compared to 233 for real data.

In this work, we set out to close the gap in fidelity and variety between images generated by GANs and real-world images from the ImageNet dataset. We make the following three contributions towards this goal:

- We demonstrate that GANs benefit dramatically from scaling, and train models with two to four times as many parameters and eight times the batch size compared to prior art. We introduce two simple, general architectural changes that improve scalability, and modify a regularization scheme to improve conditioning, demonstrably boosting performance.

[*]Work done at DeepMind
[†]Equal contribution

- As a side effect of our modifications, our models become amenable to the "truncation trick," a simple sampling technique that allows explicit, fine-grained control of the trade-off between sample variety and fidelity.
- We discover instabilities specific to large scale GANs, and characterize them empirically. Leveraging insights from this analysis, we demonstrate that a combination of novel and existing techniques can reduce these instabilities, but complete training stability can only be achieved at a dramatic cost to performance.

Our modifications substantially improve class-conditional GANs. When trained on ImageNet at 128×128 resolution, our models (BigGANs) improve the state-of-the-art Inception Score (IS) and Fréchet Inception Distance (FID) from 52.52 and 18.65 to 166.5 and 7.4 respectively. We also successfully train BigGANs on ImageNet at 256×256 and 512×512 resolution, and achieve IS and FID of 232.5 and 8.1 at 256×256 and IS and FID of 241.5 and 11.5 at 512×512. Finally, we train our models on an even larger dataset – JFT-300M – and demonstrate that our design choices transfer well from ImageNet. Code and weights for our pretrained generators are publicly available [1].

2 BACKGROUND

A Generative Adversarial Network (GAN) involves Generator (**G**) and Discriminator (**D**) networks whose purpose, respectively, is to map random noise to samples and discriminate real and generated samples. Formally, the GAN objective, in its original form (Goodfellow et al., 2014) involves finding a Nash equilibrium to the following two player min-max problem:

$$\min_G \max_D \mathbb{E}_{x\sim q_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))], \tag{1}$$

where $\boldsymbol{z} \in \mathbb{R}^{d_z}$ is a latent variable drawn from distribution $p(\boldsymbol{z})$ such as $\mathcal{N}(0, I)$ or $\mathcal{U}[-1, 1]$. When applied to images, **G** and **D** are usually convolutional neural networks (Radford et al., 2016). Without auxiliary stabilization techniques, this training procedure is notoriously brittle, requiring finely-tuned hyperparameters and architectural choices to work at all.

Much recent research has accordingly focused on modifications to the vanilla GAN procedure to impart stability, drawing on a growing body of empirical and theoretical insights (Nowozin et al., 2016; Sønderby et al., 2017; Fedus et al., 2018). One line of work is focused on changing the objective function (Arjovsky et al., 2017; Mao et al., 2016; Lim & Ye, 2017; Bellemare et al., 2017; Salimans et al., 2018) to encourage convergence. Another line is focused on constraining **D** through gradient penalties (Gulrajani et al., 2017; Kodali et al., 2017; Mescheder et al., 2018) or normalization (Miyato et al., 2018), both to counteract the use of unbounded loss functions and ensure **D** provides gradients everywhere to **G**.

Of particular relevance to our work is Spectral Normalization (Miyato et al., 2018), which enforces Lipschitz continuity on **D** by normalizing its parameters with running estimates of their first singular values, inducing backwards dynamics that adaptively regularize the top singular direction. Relatedly Odena et al. (2018) analyze the condition number of the Jacobian of **G** and find that performance is dependent on **G**'s conditioning. Zhang et al. (2018) find that employing Spectral Normalization in **G** improves stability, allowing for fewer **D** steps per iteration. We extend on these analyses to gain further insight into the pathology of GAN training.

Other works focus on the choice of architecture, such as SA-GAN (Zhang et al., 2018) which adds the self-attention block from (Wang et al., 2018) to improve the ability of both **G** and **D** to model global structure. ProGAN (Karras et al., 2018) trains high-resolution GANs in the single-class setting by training a single model across a sequence of increasing resolutions.

In conditional GANs (Mirza & Osindero, 2014) class information can be fed into the model in various ways. In (Odena et al., 2017) it is provided to **G** by concatenating a 1-hot class vector to the noise vector, and the objective is modified to encourage conditional samples to maximize the corresponding class probability predicted by an auxiliary classifier. de Vries et al. (2017) and

[1] https://tfhub.dev/s?q=biggan

Batch	Ch.	Param (M)	Shared	Skip-z	Ortho.	Itr $\times 10^3$	FID	IS
256	64	81.5	SA-GAN Baseline			1000	18.65	52.52
512	64	81.5	✗	✗	✗	1000	15.30	58.77($\pm$1.18)
1024	64	81.5	✗	✗	✗	1000	14.88	63.03($\pm$1.42)
2048	64	81.5	✗	✗	✗	732	12.39	76.85($\pm$3.83)
2048	96	173.5	✗	✗	✗	295($\pm$18)	9.54($\pm$0.62)	92.98($\pm$4.27)
2048	96	160.6	✓	✗	✗	185($\pm$11)	9.18($\pm$0.13)	94.94($\pm$1.32)
2048	96	158.3	✓	✓	✗	152($\pm$7)	8.73($\pm$0.45)	98.76($\pm$2.84)
2048	96	158.3	✓	✓	✓	165($\pm$13)	8.51($\pm$0.32)	99.31($\pm$2.10)
2048	64	71.3	✓	✓	✓	371($\pm$7)	10.48($\pm$0.10)	86.90($\pm$0.61)

Table 1: Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for ablations of our proposed modifications. *Batch* is batch size, *Param* is total number of parameters, *Ch.* is the channel multiplier representing the number of units in each layer, *Shared* is using shared embeddings, *Skip-z* is using skip connections from the latent to multiple layers, *Ortho.* is Orthogonal Regularization, and *Itr* indicates if the setting is stable to 10^6 iterations, or it collapses at the given iteration. Other than rows 1-4, results are computed across 8 random initializations.

Dumoulin et al. (2017) modify the way class conditioning is passed to **G** by supplying it with class-conditional gains and biases in BatchNorm (Ioffe & Szegedy, 2015) layers. In Miyato & Koyama (2018), **D** is conditioned by using the cosine similarity between its features and a set of learned class embeddings as additional evidence for distinguishing real and generated samples, effectively encouraging generation of samples whose features match a learned class prototype.

Objectively evaluating implicit generative models is difficult (Theis et al., 2015). A variety of works have proposed heuristics for measuring the sample quality of models without tractable likelihoods (Salimans et al., 2016; Heusel et al., 2017; Bińkowski et al., 2018; Wu et al., 2017). Of these, the Inception Score (IS, Salimans et al. (2016)) and Fréchet Inception Distance (FID, Heusel et al. (2017)) have become popular despite their notable flaws (Barratt & Sharma, 2018). We employ them as approximate measures of sample quality, and to enable comparison against previous work.

3 Scaling Up GANs

In this section, we explore methods for scaling up GAN training to reap the performance benefits of larger models and larger batches. As a baseline, we employ the SA-GAN architecture of Zhang et al. (2018), which uses the hinge loss (Lim & Ye, 2017; Tran et al., 2017) GAN objective. We provide class information to **G** with class-conditional BatchNorm (Dumoulin et al., 2017; de Vries et al., 2017) and to **D** with projection (Miyato & Koyama, 2018). The optimization settings follow Zhang et al. (2018) (notably employing Spectral Norm in **G**) with the modification that we halve the learning rates and take two **D** steps per **G** step. For evaluation, we employ moving averages of **G**'s weights following Karras et al. (2018); Mescheder et al. (2018); Yazc et al. (2018), with a decay of 0.9999. We use Orthogonal Initialization (Saxe et al., 2014), whereas previous works used $\mathcal{N}(0, 0.02I)$ (Radford et al., 2016) or Xavier initialization (Glorot & Bengio, 2010). Each model is trained on 128 to 512 cores of a Google TPUv3 Pod (Google, 2018), and computes BatchNorm statistics in **G** across all devices, rather than per-device as is typical. We find progressive growing (Karras et al., 2018) unnecessary even for our 512×512 models. Additional details are in Appendix C.

We begin by increasing the batch size for the baseline model, and immediately find tremendous benefits in doing so. Rows 1-4 of Table 1 show that simply increasing the batch size by a factor of 8 improves the state-of-the-art IS by 46%. We conjecture that this is a result of each batch covering more modes, providing better gradients for both networks. One notable side effect of this scaling is that our models reach better final performance in fewer iterations, but become unstable and undergo complete training collapse. We discuss the causes and ramifications of this in Section 4. For these experiments, we report scores from checkpoints saved just before collapse.

We then increase the width (number of channels) in each layer by 50%, approximately doubling the number of parameters in both models. This leads to a further IS improvement of 21%, which we posit is due to the increased capacity of the model relative to the complexity of the dataset. Doubling

(a) (b)

Figure 2: (a) The effects of increasing truncation. From left to right, the threshold is set to 2, 1, 0.5, 0.04. (b) Saturation artifacts from applying truncation to a poorly conditioned model.

the depth did not initially lead to improvement – we addressed this later in the BigGAN-deep model, which uses a different residual block structure.

We note that class embeddings c used for the conditional BatchNorm layers in **G** contain a large number of weights. Instead of having a separate layer for each embedding (Miyato et al., 2018; Zhang et al., 2018), we opt to use a shared embedding, which is linearly projected to each layer's gains and biases (Perez et al., 2018). This reduces computation and memory costs, and improves training speed (in number of iterations required to reach a given performance) by 37%. Next, we add direct skip connections (skip-z) from the noise vector z to multiple layers of **G** rather than just the initial layer. The intuition behind this design is to allow **G** to use the latent space to directly influence features at different resolutions and levels of hierarchy. In BigGAN, this is accomplished by splitting z into one chunk per resolution, and concatenating each chunk to the conditional vector c which gets projected to the BatchNorm gains and biases. In BigGAN-deep, we use an even simpler design, concatenating the entire z with the conditional vector without splitting it into chunks. Previous works (Goodfellow et al., 2014; Denton et al., 2015) have considered variants of this concept; our implementation is a minor modification of this design. Skip-z provides a modest performance improvement of around 4%, and improves training speed by a further 18%.

3.1 Trading off variety and fidelity with the Truncation Trick

Unlike models which need to backpropagate through their latents, GANs can employ an arbitrary prior $p(z)$, yet the vast majority of previous works have chosen to draw z from either $\mathcal{N}(0, I)$ or $\mathcal{U}[-1, 1]$. We question the optimality of this choice and explore alternatives in Appendix E.

Remarkably, our best results come from using a different latent distribution for sampling than was used in training. Taking a model trained with $z \sim \mathcal{N}(0, I)$ and sampling z from a *truncated normal* (where values which fall outside a range are resampled to fall inside that range) immediately provides a boost to IS and FID. We call this the *Truncation Trick*: truncating a z vector by resampling the values with magnitude above a chosen threshold leads to improvement in individual sample quality at the cost of reduction in overall sample variety. Figure 2(a) demonstrates this: as the threshold is reduced, and elements of z are truncated towards zero (the mode of the latent distribution), individual samples approach the mode of **G**'s output distribution. Related observations about this trade-off were made in (Marchesi, 2016; Pieters & Wiering, 2014).

This technique allows fine-grained, post-hoc selection of the trade-off between sample quality and variety for a given **G**. Notably, we can compute FID and IS for a range of thresholds, obtaining the variety-fidelity curve reminiscent of the precision-recall curve (Figure 17). As IS does not penalize lack of variety in class-conditional models, reducing the truncation threshold leads to a direct increase in IS (analogous to precision). FID penalizes lack of variety (analogous to recall) but also rewards precision, so we initially see a moderate improvement in FID, but as truncation approaches zero and variety diminishes, the FID sharply drops. The distribution shift caused by sampling with different latents than those seen in training is problematic for many models. Some of our larger models are not amenable to truncation, producing saturation artifacts (Figure 2(b)) when fed truncated noise. To counteract this, we seek to enforce amenability to truncation by conditioning **G** to be smooth, so that the full space of z will map to good output samples. For this, we turn to Orthogonal Regularization (Brock et al., 2017), which directly enforces the orthogonality condition:

$$R_\beta(W) = \beta \|W^\top W - I\|_{\mathrm{F}}^2, \tag{2}$$

where W is a weight matrix and β a hyperparameter. This regularization is known to often be too limiting (Miyato et al., 2018), so we explore several variants designed to relax the constraint while still imparting the desired smoothness to our models. The version we find to work best removes the diagonal terms from the regularization, and aims to minimize the pairwise cosine similarity between filters but does not constrain their norm:

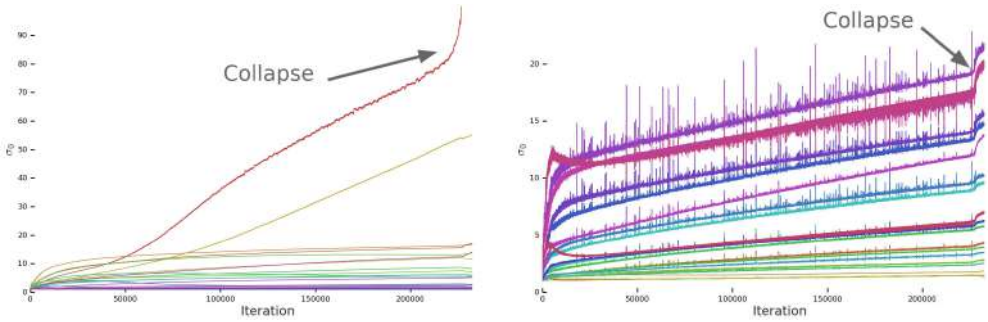
$$R_\beta(W) = \beta \|W^\top W \odot (\mathbf{1} - I)\|_{\mathrm{F}}^2, \tag{3}$$

where $\mathbf{1}$ denotes a matrix with all elements set to 1. We sweep β values and select 10^{-4}, finding this small added penalty sufficient to improve the likelihood that our models will be amenable to truncation. Across runs in Table 1, we observe that without Orthogonal Regularization, only 16% of models are amenable to truncation, compared to 60% when trained with Orthogonal Regularization.

3.2 Summary

We find that current GAN techniques are sufficient to enable scaling to large models and distributed, large-batch training. We find that we can dramatically improve the state of the art and train models up to 512×512 resolution without need for explicit multiscale methods like Karras et al. (2018). Despite these improvements, our models undergo training collapse, necessitating early stopping in practice. In the next two sections we investigate why settings which were stable in previous works become unstable when applied at scale.

4 Analysis



(a) **G** (b) **D**

Figure 3: A typical plot of the first singular value σ_0 in the layers of **G** (a) and **D** (b) before Spectral Normalization. Most layers in **G** have well-behaved spectra, but without constraints a small subset grow throughout training and explode at collapse. **D**'s spectra are noisier but otherwise better-behaved. Colors from red to violet indicate increasing depth.

4.1 Characterizing Instability: The Generator

Much previous work has investigated GAN stability from a variety of analytical angles and on toy problems, but the instabilities we observe occur for settings which are stable at small scale, necessitating direct analysis at large scale. We monitor a range of weight, gradient, and loss statistics during training, in search of a metric which might presage the onset of training collapse, similar to (Odena et al., 2018). We found the top three singular values $\sigma_0, \sigma_1, \sigma_2$ of each weight matrix to be the most informative. They can be efficiently computed using the Alrnoldi iteration method (Golub & der Vorst, 2000), which extends the power iteration method, used in Miyato et al. (2018), to estimation of additional singular vectors and values. A clear pattern emerges, as can be seen in Figure 3(a) and Appendix F: most **G** layers have well-behaved spectral norms, but some layers

(typically the first layer in **G**, which is over-complete and not convolutional) are ill-behaved, with spectral norms that grow throughout training and explode at collapse.

To ascertain if this pathology is a cause of collapse or merely a symptom, we study the effects of imposing additional conditioning on **G** to explicitly counteract spectral explosion. First, we directly regularize the top singular values σ_0 of each weight, either towards a fixed value σ_{reg} or towards some ratio r of the second singular value, $r \cdot sg(\sigma_1)$ (with sg the stop-gradient operation to prevent the regularization from increasing σ_1). Alternatively, we employ a partial singular value decomposition to instead clamp σ_0. Given a weight W, its first singular vectors u_0 and v_0, and σ_{clamp} the value to which the σ_0 will be clamped, our weights become:

$$W = W - \max(0, \sigma_0 - \sigma_{clamp}) v_0 u_0^\top, \tag{4}$$

where σ_{clamp} is set to either σ_{reg} or $r \cdot sg(\sigma_1)$. We observe that both with and without Spectral Normalization these techniques have the effect of preventing the gradual increase and explosion of either σ_0 or $\frac{\sigma_0}{\sigma_1}$, but even though in some cases they mildly improve performance, no combination prevents training collapse. This evidence suggests that while conditioning **G** might improve stability, it is insufficient to ensure stability. We accordingly turn our attention to **D**.

4.2 Characterizing Instability: The Discriminator

As with **G**, we analyze the spectra of **D**'s weights to gain insight into its behavior, then seek to stabilize training by imposing additional constraints. Figure 3(b) displays a typical plot of σ_0 for **D** (with further plots in Appendix F). Unlike **G**, we see that the spectra are noisy, $\frac{\sigma_0}{\sigma_1}$ is well-behaved, and the singular values grow throughout training but only jump at collapse, instead of exploding.

The spikes in **D**'s spectra might suggest that it periodically receives very large gradients, but we observe that the Frobenius norms are smooth (Appendix F), suggesting that this effect is primarily concentrated on the top few singular directions. We posit that this noise is a result of optimization through the adversarial training process, where **G** periodically produces batches which strongly perturb **D** . If this spectral noise is causally related to instability, a natural counter is to employ gradient penalties, which explicitly regularize changes in **D**'s Jacobian. We explore the R_1 zero-centered gradient penalty from Mescheder et al. (2018):

$$R_1 := \frac{\gamma}{2} \mathbb{E}_{p_{\mathcal{D}}(x)} \left[\|\nabla D(x)\|_F^2 \right]. \tag{5}$$

With the default suggested γ strength of 10, training becomes stable and improves the smoothness and boundedness of spectra in both **G** and **D**, but performance severely degrades, resulting in a 45% reduction in IS. Reducing the penalty partially alleviates this degradation, but results in increasingly ill-behaved spectra; even with the penalty strength reduced to 1 (the lowest strength for which sudden collapse does not occur) the IS is reduced by 20%. Repeating this experiment with various strengths of Orthogonal Regularization, DropOut (Srivastava et al., 2014), and L2 (See Appendix I for details), reveals similar behaviors for these regularization strategies: with high enough penalties on **D**, training stability can be achieved, but at a substantial cost to performance.

We also observe that **D**'s loss approaches zero during training, but undergoes a sharp upward jump at collapse (Appendix F). One possible explanation for this behavior is that **D** is overfitting to the training set, memorizing training examples rather than learning some meaningful boundary between real and generated images. As a simple test for **D**'s memorization (related to Gulrajani et al. (2017)), we evaluate uncollapsed discriminators on the ImageNet training and validation sets, and measure what percentage of samples are classified as real or generated. While the training accuracy is consistently above 98%, the validation accuracy falls in the range of 50-55%, no better than random guessing (regardless of regularization strategy). This confirms that **D** is indeed memorizing the training set; we deem this in line with **D**'s role, which is not explicitly to generalize, but to distill the training data and provide a useful learning signal for **G**. Additional experiments and discussion are provided in Appendix G.

4.3 Summary

We find that stability does not come solely from **G** or **D**, but from their interaction through the adversarial training process. While the symptoms of their poor conditioning can be used to track and

Model	Res.	FID/IS	(min FID) / IS	FID / (valid IS)	FID / (max IS)
SN-GAN	128	$27.62/36.80$	N/A	N/A	N/A
SA-GAN	128	$18.65/52.52$	N/A	N/A	N/A
BigGAN	128	$8.7 \pm .6/98.8 \pm 3$	$7.7 \pm .2/126.5 \pm 0$	$9.6 \pm .4/166.3 \pm 1$	$25 \pm 2/206 \pm 2$
BigGAN	256	$8.7 \pm .1/142.3 \pm 2$	$7.7 \pm .1/178.0 \pm 5$	$9.3 \pm .3/233.1 \pm 1$	$25 \pm 5/291 \pm 4$
BigGAN	512	$8.1/144.2$	$7.6/170.3$	$11.8/241.4$	$27.0/275$
BigGAN-deep	128	$5.7 \pm .3/124.5 \pm 2$	$6.3 \pm .3/148.1 \pm 4$	$7.4 \pm .6/166.5 \pm 1$	$25 \pm 2/253 \pm 11$
BigGAN-deep	256	$6.9 \pm .2/171.4 \pm 2$	$7.0 \pm .1/202.6 \pm 2$	$8.1 \pm .1/232.5 \pm 2$	$27 \pm 8/317 \pm 6$
BigGAN-deep	512	$7.5/152.8$	$7.7/181.4$	$11.5/241.5$	$39.7/298$

Table 2: Evaluation of models at different resolutions. We report scores without truncation (Column 3), scores at the best FID (Column 4), scores at the IS of validation data (Column 5), and scores at the max IS (Column 6). Standard deviations are computed over at least three random initializations.

identify instability, ensuring reasonable conditioning proves necessary for training but insufficient to prevent eventual training collapse. It is possible to enforce stability by strongly constraining `D`, but doing so incurs a dramatic cost in performance. With current techniques, better final performance can be achieved by relaxing this conditioning and allowing collapse to occur at the later stages of training, by which time a model is sufficiently trained to achieve good results.

5 Experiments

(a) 128×128 (b) 256×256 (c) 512×512 (d)

Figure 4: Samples from our BigGAN model with truncation threshold 0.5 (a-c) and an example of class leakage in a partially trained model (d).

5.1 Evaluation on ImageNet

We evaluate our models on ImageNet ILSVRC 2012 (Russakovsky et al., 2015) at 128×128, 256×256, and 512×512 resolutions, employing the settings from Table 1, row 8. The samples generated by our models are presented in Figure 4, with additional samples in Appendix A, and online [2]. We report IS and FID in Table 2. As our models are able to trade sample variety for quality, it is unclear how best to compare against prior art; we accordingly report values at three settings, with complete curves in Appendix D. First, we report the FID/IS values at the truncation setting which attains the best FID. Second, we report the FID at the truncation setting for which our model's IS is the same as that attained by the real validation data, reasoning that this is a passable measure of maximum sample variety achieved while still achieving a good level of "objectness." Third, we report FID at the maximum IS achieved by each model, to demonstrate how much variety must be traded off to maximize quality. In all three cases, our models outperform the previous state-of-the-art IS and FID scores achieved by Miyato et al. (2018) and Zhang et al. (2018).

In addition to the BigGAN model introduced in the first version of the paper and used in the majority of experiments (unless otherwise stated), we also present a 4x deeper model (BigGAN-deep) which uses a different configuration of residual blocks. As can be seen from Table 2, BigGAN-deep substantially outperforms BigGAN across all resolutions and metrics. This confirms that our findings

[2] https://drive.google.com/drive/folders/1lWC6XEPD0LT5KUnPXeve_kWeY-FxH002

Ch.	Param (M)	Shared	Skip-z	Ortho.	FID	IS	(min FID) / IS	FID / (max IS)
64	317.1	✗	✗	✗	48.38	23.27	48.6/23.1	49.1/23.9
64	99.4	✓	✓	✓	23.48	24.78	22.4/21.0	60.9/35.8
96	207.9	✓	✓	✓	18.84	27.86	17.1/23.3	51.6/38.1
128	355.7	✓	✓	✓	13.75	30.61	13.0/28.0	46.2/47.8

Table 3: BigGAN results on JFT-300M at 256×256 resolution. The *FID* and *IS* columns report these scores given by the JFT-300M-trained Inception v2 classifier with noise distributed as $z \sim \mathcal{N}(0, I)$ (non-truncated). The *(min FID) / IS* and *FID / (max IS)* columns report scores at the best FID and IS from a sweep across truncated noise distributions ranging from $\sigma = 0$ to $\sigma = 2$. Images from the JFT-300M validation set have an IS of 50.88 and FID of 1.94.

extend to other architectures, and that increased depth leads to improvement in sample quality. Both BigGAN and BigGAN-deep architectures are described in Appendix B.

Our observation that **D** overfits to the training set, coupled with our model's sample quality, raises the obvious question of whether or not **G** simply memorizes training points. To test this, we perform class-wise nearest neighbors analysis in pixel space and the feature space of pre-trained classifier networks (Appendix A). In addition, we present both interpolations between samples and class-wise interpolations (where z is held constant) in Figures 8 and 9. Our model convincingly interpolates between disparate samples, and the nearest neighbors for its samples are visually distinct, suggesting that our model does not simply memorize training data.

We note that some failure modes of our partially-trained models are distinct from those previously observed. Most previous failures involve local artifacts (Odena et al., 2016), images consisting of texture blobs instead of objects (Salimans et al., 2016), or the canonical mode collapse. We observe *class leakage*, where images from one class contain properties of another, as exemplified by Figure 4(d). We also find that many classes on ImageNet are more difficult than others for our model; our model is more successful at generating dogs (which make up a large portion of the dataset, and are mostly distinguished by their texture) than crowds (which comprise a small portion of the dataset and have more large-scale structure). Further discussion is available in Appendix A.

5.2 Additional evaluation on JFT-300M

To confirm that our design choices are effective for even larger and more complex and diverse datasets, we also present results of our system on a subset of JFT-300M (Sun et al., 2017). The full JFT-300M dataset contains 300M real-world images labeled with 18K categories. Since the category distribution is heavily long-tailed, we subsample the dataset to keep only images with the 8.5K most common labels. The resulting dataset contains 292M images – two orders of magnitude larger than ImageNet. For images with multiple labels, we sample a single label randomly and independently whenever an image is sampled. To compute IS and FID for the GANs trained on this dataset, we use an Inception v2 classifier (Szegedy et al., 2016) trained on this dataset. Quantitative results are presented in Table 3. All models are trained with batch size 2048. We compare an ablated version of our model – comparable to SA-GAN (Zhang et al., 2018) but with the larger batch size – against a "full" BigGAN model that makes uses of all of the techniques applied to obtain the best results on ImageNet (shared embedding, skip-z, and orthogonal regularization). Our results show that these techniques substantially improve performance even in the setting of this much larger dataset at the same model capacity (64 base channels). We further show that for a dataset of this scale, we see significant additional improvements from expanding the capacity of our models to 128 base channels, while for ImageNet GANs that additional capacity was not beneficial.

In Figure 19 (Appendix D), we present truncation plots for models trained on this dataset. Unlike for ImageNet, where truncation limits of $\sigma \approx 0$ tend to produce the highest fidelity scores, IS is typically maximized for our JFT-300M models when the truncation value σ ranges from 0.5 to 1. We suspect that this is at least partially due to the intra-class variability of JFT-300M labels, as well as the relative complexity of the image distribution, which includes images with multiple objects at a variety of scales. Interestingly, unlike models trained on ImageNet, where training tends to collapse without heavy regularization (Section 4), the models trained on JFT-300M remain stable over many

hundreds of thousands of iterations. This suggests that moving beyond ImageNet to larger datasets may partially alleviate GAN stability issues.

The improvement over the baseline GAN model that we achieve on this dataset without changes to the underlying models or training and regularization techniques (beyond expanded capacity) demonstrates that our findings extend from ImageNet to datasets with scale and complexity thus far unprecedented for generative models of images.

6 Conclusion

We have demonstrated that Generative Adversarial Networks trained to model natural images of multiple categories highly benefit from scaling up, both in terms of fidelity and variety of the generated samples. As a result, our models set a new level of performance among ImageNet GAN models, improving on the state of the art by a large margin. We have also presented an analysis of the training behavior of large scale GANs, characterized their stability in terms of the singular values of their weights, and discussed the interplay between stability and performance.

Acknowledgments

We would like to thank Kai Arulkumaran, Matthias Bauer, Peter Buchlovsky, Jeffrey Defauw, Sander Dieleman, Ian Goodfellow, Ariel Gordon, Karol Gregor, Dominik Grewe, Chris Jones, Jacob Menick, Augustus Odena, Suman Ravuri, Ali Razavi, Mihaela Rosca, and Jeff Stanway.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *OSDI*, 2016.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

Shane Barratt and Rishi Sharma. A note on the Inception Score. In *arXiv preprint arXiv:1801.01973*, 2018.

Marc G. Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The Cramer distance as a solution to biased Wasserstein gradients. In *arXiv preprint arXiv:1705.10743*, 2017.

Mikolaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *ICLR*, 2018.

Andrew Brock, Theodore Lim, J.M. Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *ICLR*, 2017.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016.

Harm de Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville. Modulating early visual processing by language. In *NIPS*, 2017.

Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2017.

William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M. Dai, Shakir Mohamed, and Ian Goodfellow. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. In *ICLR*, 2018.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

Gene Golub and Henk Van der Vorst. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, 123:35–65, 2000.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, and Aaron Courville Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

Google. Cloud TPUs. `https://cloud.google.com/tpu/`, 2018.

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of Wasserstein GANs. In *NIPS*, 2017.

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

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of GANs. In *arXiv preprint arXiv:1705.07215*, 2017.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Jae Hyun Lim and Jong Chul Ye. Geometric GAN. In *arXiv preprint arXiv:1705.02894*, 2017.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Least squares generative adversarial networks. In *arXiv preprint arXiv:1611.04076*, 2016.

Marco Marchesi. Megapixel size image creation using generative adversarial networks. In *arXiv preprint arXiv:1706.00082*, 2016.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *ICML*, 2018.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. In *arXiv preprint arXiv:1411.1784*, 2014.

Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. In *ICLR*, 2018.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *NIPS*, 2016.

Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *ICML*, 2017.

Augustus Odena, Jacob Buckman, Catherine Olsson, Tom B. Brown, Christopher Olah, Colin Raffel, and Ian Goodfellow. Is generator conditioning causally related to GAN performance? In *ICML*, 2018.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Mathijs Pieters and Marco Wiering. Comparing generative adversarial network techniques for image creation and modificatio. In *arXiv preprint arXiv:1803.09093*, 2014.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, and Michael Bernstein. ImageNet large scale visual recognition challenge. *IJCV*, 115:211–252, 2015.

Tim Salimans and Diederik Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *NIPS*, 2016.

Tim Salimans, Han Zhang, Alec Radford, and Dimitris Metaxas. Improving GANs using optimal transport. In *ICLR*, 2018.

Andrew Saxe, James McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*, 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszr. Amortised map inference for image super-resolution. In *ICLR*, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:1929–1958, 2014.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. In *arXiv preprint arXiv:1511.01844*, 2015.

Dustin Tran, Rajesh Ranganath, and David M. Blei. Hierarchical implicit models and likelihood-free variational inference. In *NIPS*, 2017.

Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger B. Grosse. On the quantitative analysis of decoder-based generative models. In *ICLR*, 2017.

Yasin Yazc, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in gan training. In *arXiv preprint arXiv:1806.04498*, 2018.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *arXiv preprint arXiv:1805.08318*, 2018.

APPENDIX A ADDITIONAL SAMPLES, INTERPOLATIONS, AND NEAREST NEIGHBORS FROM IMAGENET MODELS

Figure 5: Samples generated by our BigGAN model at 256×256 resolution.

Figure 6: Samples generated by our BigGAN model at 512×512 resolution.

(a) (b)

Figure 7: Comparing easy classes (a) with difficult classes (b) at 512×512. Classes such as dogs which are largely textural, and common in the dataset, are far easier to model than classes involving unaligned human faces or crowds. Such classes are more dynamic and structured, and often have details to which human observers are more sensitive. The difficulty of modeling global structure is further exacerbated when producing high-resolution images, even with non-local blocks.

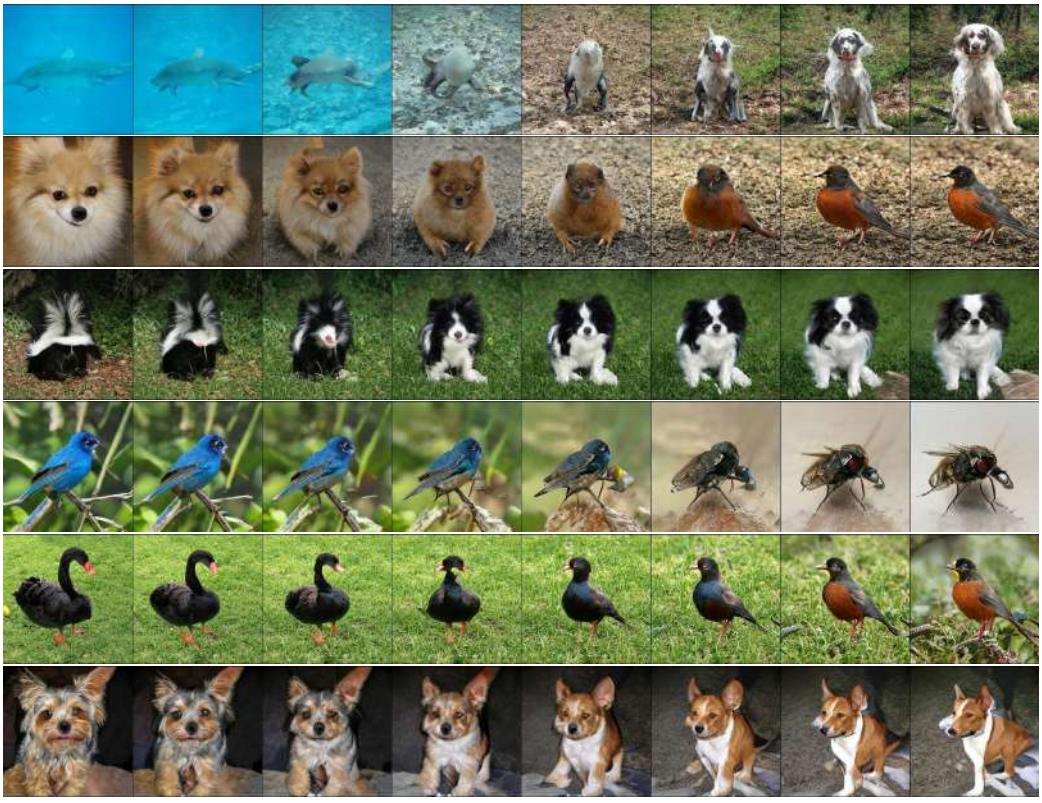


Figure 8: Interpolations between z, c pairs.

Figure 9: Interpolations between c with z held constant. Pose semantics are frequently maintained between endpoints (particularly in the final row). Row 2 demonstrates that grayscale is encoded in the joint z, c space, rather than in z.

Figure 10: Nearest neighbors in VGG-16-fc7 (Simonyan & Zisserman, 2015) feature space. The generated image is in the top left.

Figure 11: Nearest neighbors in ResNet-50-avgpool (He et al., 2016) feature space. The generated image is in the top left.

Figure 12: Nearest neighbors in pixel space. The generated image is in the top left.

Figure 13: Nearest neighbors in VGG-16-fc7 (Simonyan & Zisserman, 2015) feature space. The generated image is in the top left.

Figure 14: Nearest neighbors in ResNet-50-avgpool (He et al., 2016) feature space. The generated image is in the top left.

APPENDIX B ARCHITECTURAL DETAILS

In the BigGAN model (Figure 15), we use the ResNet (He et al., 2016) GAN architecture of (Zhang et al., 2018), which is identical to that used by (Miyato et al., 2018), but with the channel pattern in **D** modified so that the number of filters in the first convolutional layer of each block is equal to the number of output filters (rather than the number of input filters, as in Miyato et al. (2018); Gulrajani et al. (2017)). We use a single shared class embedding in **G**, and skip connections for the latent vector z (skip-z). In particular, we employ hierarchical latent spaces, so that the latent vector z is split along its channel dimension into chunks of equal size (20-D in our case), and each chunk is concatenated to the shared class embedding and passed to a corresponding residual block as a conditioning vector. The conditioning of each block is linearly projected to produce per-sample gains and biases for the BatchNorm layers of the block. The bias projections are zero-centered, while the gain projections are centered at 1. Since the number of residual blocks depends on the image resolution, the full dimensionality of z is 120 for 128×128, 140 for 256×256, and 160 for 512×512 images.

The BigGAN-deep model (Figure 16) differs from BigGAN in several aspects. It uses a simpler variant of skip-z conditioning: instead of first splitting z into chunks, we concatenate the entire z with the class embedding, and pass the resulting vector to each residual block through skip connections. BigGAN-deep is based on residual blocks with bottlenecks (He et al., 2016), which incorporate two additional 1×1 convolutions: the first reduces the number of channels by a factor of 4 before the more expensive 3×3 convolutions; the second produces the required number of output channels. While BigGAN relies on 1×1 convolutions in the skip connections whenever the number of channels needs to change, in BigGAN-deep we use a different strategy aimed at preserving identity throughout the skip connections. In **G**, where the number of channels needs to be reduced, we simply retain the first group of channels and drop the rest to produce the required number of channels. In **D**, where the number of channels should be increased, we pass the input channels unperturbed, and concatenate them with the remaining channels produced by a 1×1 convolution. As far as the network configuration is concerned, the discriminator is an exact reflection of the generator. There are two blocks at each resolution (BigGAN uses one), and as a result BigGAN-deep is four times deeper than BigGAN. Despite their increased depth, the BigGAN-deep models have significantly fewer parameters mainly due to the bottleneck structure of their residual blocks. For example, the 128×128 BigGAN-deep **G** and **D** have 50.4M and 34.6M parameters respectively, while the corresponding original BigGAN models have 70.4M and 88.0M parameters. All BigGAN-deep models use attention at 64×64 resolution, channel width multiplier $ch = 128$, and $z \in \mathbb{R}^{128}$.

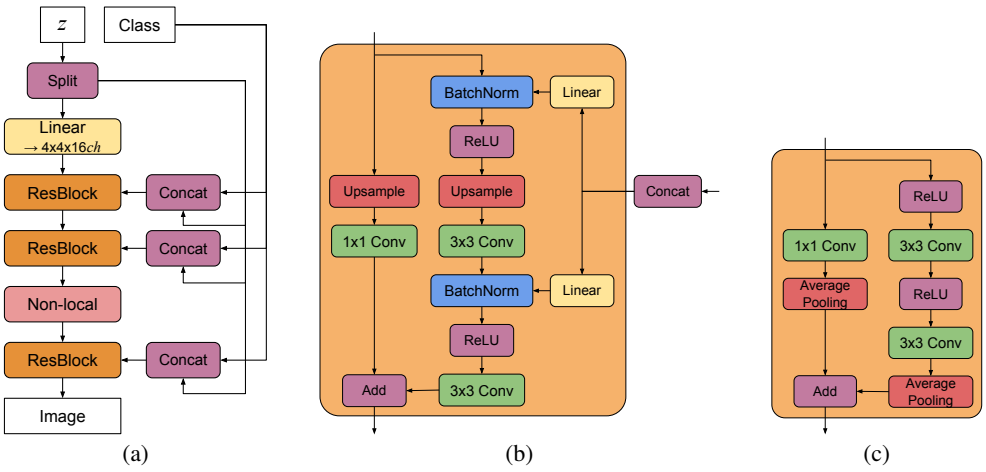


Figure 15: (a) A typical architectural layout for BigGAN's **G**; details are in the following tables. (b) A Residual Block (*ResBlock up*) in BigGAN's **G**. (c) A Residual Block (*ResBlock down*) in BigGAN's **D**.

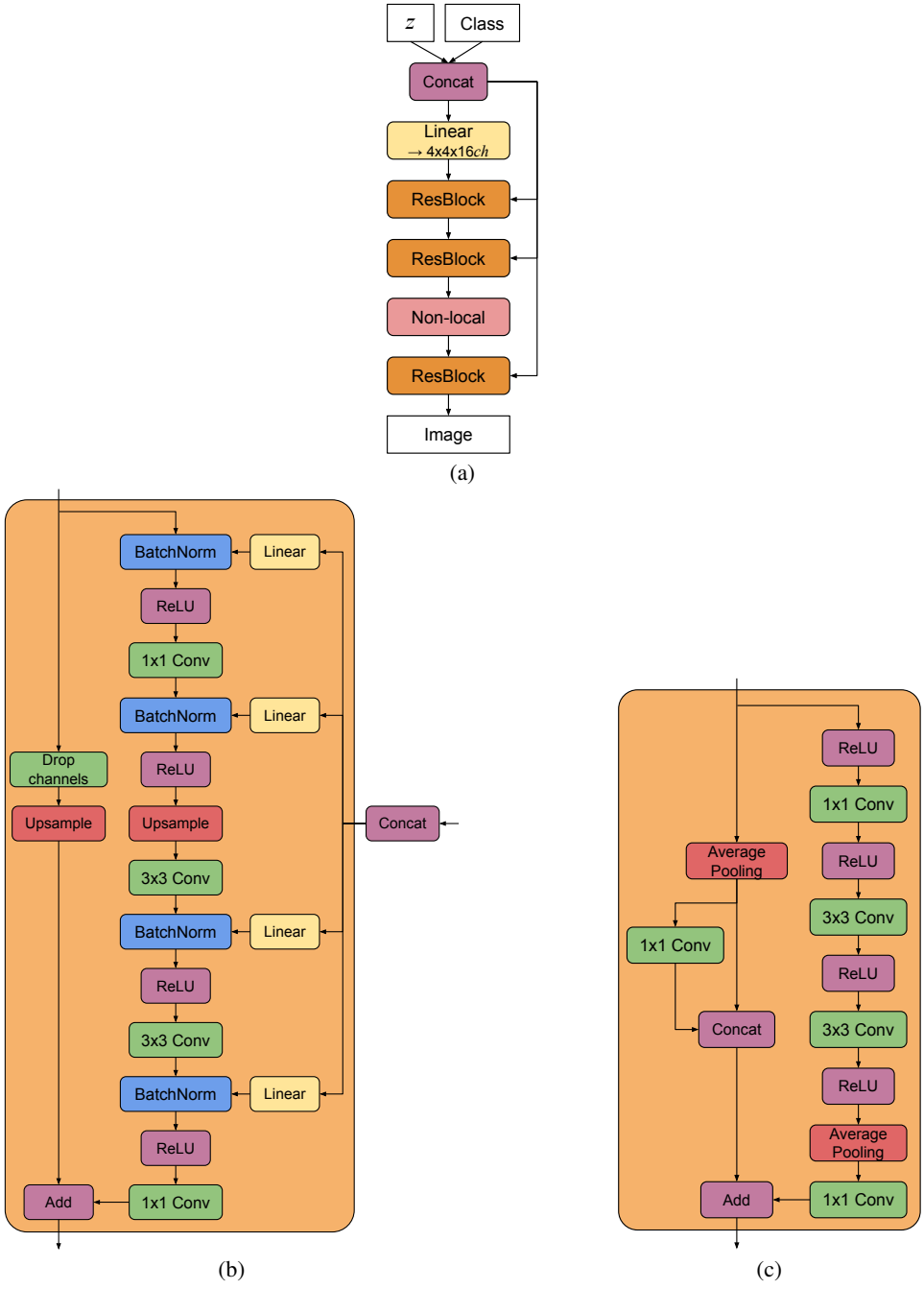


Figure 16: (a) A typical architectural layout for BigGAN-deep's **G**; details are in the following tables. (b) A Residual Block (*ResBlock up*) in BigGAN-deep's **G**. (c) A Residual Block (*ResBlock down*) in BigGAN-deep's **D**. A *ResBlock* (without *up* or *down*) in BigGAN-deep does not include the *Upsample* or *Average Pooling* layers, and has identity skip connections.

Table 4: BigGAN architecture for 128×128 images. ch represents the channel width multiplier in each network from Table 1.

$z \in \mathbb{R}^{120} \sim \mathcal{N}(0, I)$ Embed$(y) \in \mathbb{R}^{128}$
Linear $(20 + 128) \rightarrow 4 \times 4 \times 16ch$
ResBlock up $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 4ch$
ResBlock up $4ch \rightarrow 2ch$
Non-Local Block (64×64)
ResBlock up $2ch \rightarrow ch$
BN, ReLU, 3×3 Conv $ch \rightarrow 3$
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{128 \times 128 \times 3}$
ResBlock down $ch \rightarrow 2ch$
Non-Local Block (64×64)
ResBlock down $2ch \rightarrow 4ch$
ResBlock down $4ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$
ResBlock down $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ReLU, Global sum pooling
Embed$(y) \cdot \boldsymbol{h}$ + (linear $\rightarrow$ 1)

(b) Discriminator

Table 5: BigGAN architecture for 256×256 images. Relative to the 128×128 architecture, we add an additional ResBlock in each network at 16×16 resolution, and move the non-local block in **G** to 128×128 resolution. Memory constraints prevent us from moving the non-local block in **D**.

$z \in \mathbb{R}^{140} \sim \mathcal{N}(0, I)$ Embed$(y) \in \mathbb{R}^{128}$
Linear $(20 + 128) \rightarrow 4 \times 4 \times 16ch$
ResBlock up $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 4ch$
ResBlock up $4ch \rightarrow 2ch$
Non-Local Block (128×128)
ResBlock up $2ch \rightarrow ch$
BN, ReLU, 3×3 Conv $ch \rightarrow 3$
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{256 \times 256 \times 3}$
ResBlock down $ch \rightarrow 2ch$
ResBlock down $2ch \rightarrow 4ch$
Non-Local Block (64×64)
ResBlock down $4ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$
ResBlock down $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ReLU, Global sum pooling
Embed$(y) \cdot \boldsymbol{h}$ + (linear $\rightarrow$ 1)

(b) Discriminator

Table 6: BigGAN architecture for 512×512 images. Relative to the 256×256 architecture, we add an additional ResBlock at the 512×512 resolution. Memory constraints force us to move the non-local block in both networks back to 64×64 resolution as in the 128×128 pixel setting.

$z \in \mathbb{R}^{160} \sim \mathcal{N}(0, I)$ Embed$(y) \in \mathbb{R}^{128}$
Linear $(20 + 128) \to 4 \times 4 \times 16ch$
ResBlock up $16ch \to 16ch$
ResBlock up $16ch \to 8ch$
ResBlock up $8ch \to 8ch$
ResBlock up $8ch \to 4ch$
Non-Local Block (64×64)
ResBlock up $4ch \to 2ch$
ResBlock up $2ch \to ch$
ResBlock up $ch \to ch$
BN, ReLU, 3×3 Conv $ch \to 3$
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{512 \times 512 \times 3}$
ResBlock down $ch \to ch$
ResBlock down $ch \to 2ch$
ResBlock down $2ch \to 4ch$
Non-Local Block (64×64)
ResBlock down $4ch \to 8ch$
ResBlock down $8ch \to 8ch$
ResBlock down $8ch \to 16ch$
ResBlock down $16ch \to 16ch$
ResBlock $16ch \to 16ch$
ReLU, Global sum pooling
Embed$(y) \cdot \boldsymbol{h}$ + (linear $\to 1$)

(b) Discriminator

Table 7: BigGAN-deep architecture for 128×128 images.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$ Embed$(y) \in \mathbb{R}^{128}$
Linear $(128 + 128) \to 4 \times 4 \times 16ch$
ResBlock $16ch \to 16ch$
ResBlock up $16ch \to 16ch$
ResBlock $16ch \to 16ch$
ResBlock up $16ch \to 8ch$
ResBlock $8ch \to 8ch$
ResBlock up $8ch \to 4ch$
ResBlock $4ch \to 4ch$
ResBlock up $4ch \to 2ch$
Non-Local Block (64×64)
ResBlock $2ch \to 2ch$
ResBlock up $2ch \to ch$
BN, ReLU, 3×3 Conv $ch \to 3$
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{128 \times 128 \times 3}$
3×3 Conv $3 \to ch$
ResBlock down $ch \to 2ch$
ResBlock $2ch \to 2ch$
Non-Local Block (64×64)
ResBlock down $2ch \to 4ch$
ResBlock $4ch \to 4ch$
ResBlock down $4ch \to 8ch$
ResBlock $8ch \to 8ch$
ResBlock down $8ch \to 16ch$
ResBlock $16ch \to 16ch$
ResBlock down $16ch \to 16ch$
ResBlock $16ch \to 16ch$
ReLU, Global sum pooling
Embed$(y) \cdot \boldsymbol{h}$ + (linear $\to 1$)

(b) Discriminator

Table 8: BigGAN-deep architecture for 256×256 images.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$ Embed(y) $\in \mathbb{R}^{128}$
Linear $(128 + 128) \rightarrow 4 \times 4 \times 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 4ch$
Non-Local Block (64×64)
ResBlock $4ch \rightarrow 4ch$
ResBlock up $4ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock up $2ch \rightarrow ch$
BN, ReLU, 3×3 Conv $ch \rightarrow 3$
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{256 \times 256 \times 3}$
3×3 Conv $3 \rightarrow ch$
ResBlock down $ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock down $2ch \rightarrow 4ch$
ResBlock $4ch \rightarrow 4ch$
Non-Local Block (64×64)
ResBlock down $4ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock down $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ReLU, Global sum pooling
Embed(y)$\cdot \boldsymbol{h}$ + (linear $\rightarrow$ 1)

(b) Discriminator

Table 9: BigGAN-deep architecture for 512×512 images.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$ Embed$(y) \in \mathbb{R}^{128}$
Linear $(128 + 128) \rightarrow 4 \times 4 \times 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 4ch$
Non-Local Block (64×64)
ResBlock $4ch \rightarrow 4ch$
ResBlock up $4ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock up $2ch \rightarrow ch$
ResBlock $ch \rightarrow ch$
ResBlock up $ch \rightarrow ch$
BN, ReLU, 3×3 Conv $ch \rightarrow 3$
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{512 \times 512 \times 3}$
3×3 Conv $3 \rightarrow ch$
ResBlock down $ch \rightarrow ch$
ResBlock $ch \rightarrow ch$
ResBlock down $ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock down $2ch \rightarrow 4ch$
ResBlock $4ch \rightarrow 4ch$
Non-Local Block (64×64)
ResBlock down $4ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock down $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ReLU, Global sum pooling
Embed$(y) \cdot \boldsymbol{h}$ + (linear $\rightarrow$ 1)

(b) Discriminator

APPENDIX C EXPERIMENTAL DETAILS

Our basic setup follows SA-GAN (Zhang et al., 2018), and is implemented in TensorFlow (Abadi et al., 2016). We employ the architectures detailed in Appendix B, with non-local blocks inserted at a single stage in each network. Both **G** and **D** networks are initialized with Orthogonal Initialization (Saxe et al., 2014). We use Adam optimizer (Kingma & Ba, 2014) with $\beta_1 = 0$ and $\beta_2 = 0.999$ and a constant learning rate. For BigGAN models at all resolutions, we use $2 \cdot 10^{-4}$ in **D** and $5 \cdot 10^{-5}$ in **G**. For BigGAN-deep, we use the learning rate of $2 \cdot 10^{-4}$ in **D** and $5 \cdot 10^{-5}$ in **G** for 128×128 models, and $2.5 \cdot 10^{-5}$ in both **D** and **G** for 256×256 and 512×512 models. We experimented with the number of **D** steps per **G** step (varying it from 1 to 6) and found that two **D** steps per **G** step gave the best results.

We use an exponential moving average of the weights of **G** at sampling time, with a decay rate set to 0.9999. We employ cross-replica BatchNorm (Ioffe & Szegedy, 2015) in **G**, where batch statistics are aggregated across all devices, rather than a single device as in standard implementations. Spectral Normalization (Miyato et al., 2018) is used in both **G** and **D**, following SA-GAN (Zhang et al., 2018). We train on a Google TPU v3 Pod, with the number of cores proportional to the resolution: 128 for 128×128, 256 for 256×256, and 512 for 512×512. Training takes between 24 and 48 hours for most models. We increase ϵ from the default 10^{-8} to 10^{-4} in BatchNorm and Spectral Norm to mollify low-precision numerical issues. We preprocess data by cropping along the long edge and rescaling to a given resolution with area resampling.

C.1 BATCHNORM STATISTICS AND SAMPLING

The default behavior with batch normalized classifier networks is to use a running average of the activation moments at test time. Previous works (Radford et al., 2016) have instead used batch statistics when sampling images. While this is not technically an invalid way to sample, it means that results are dependent on the test batch size (and how many devices it is split across), and further complicates reproducibility.

We find that this detail is extremely important, with changes in test batch size producing drastic changes in performance. This is further exacerbated when one uses exponential moving averages of **G**'s weights for sampling, as the BatchNorm running averages are computed with non-averaged weights and are poor estimates of the activation statistics for the averaged weights.

To counteract both these issues, we employ "standing statistics," where we compute activation statistics at sampling time by running the **G** through multiple forward passes (typically 100) each with different batches of random noise, and storing means and variances aggregated across all forward passes. Analogous to using running statistics, this results in **G**'s outputs becoming invariant to batch size and the number of devices, even when producing a single sample.

C.2 CIFAR-10

We run our networks on CIFAR-10 (Krizhevsky & Hinton, 2009) using the settings from Table 1, row 8, and achieve an IS of 9.22 and an FID of 14.73 without truncation.

C.3 INCEPTION SCORES OF IMAGENET IMAGES

We compute the IS for both the training and validation sets of ImageNet. At 128×128 the training data has an IS of 233, and the validation data has an IS of 166. At 256×256 the training data has an IS of 377, and the validation data has an IS of 234. At 512×512 the training data has an IS of 348, and the validation data has an IS of 241. The discrepancy between training and validation scores is due to the Inception classifier having been trained on the training data, resulting in high-confidence outputs that are preferred by the Inception Score.

APPENDIX D ADDITIONAL PLOTS

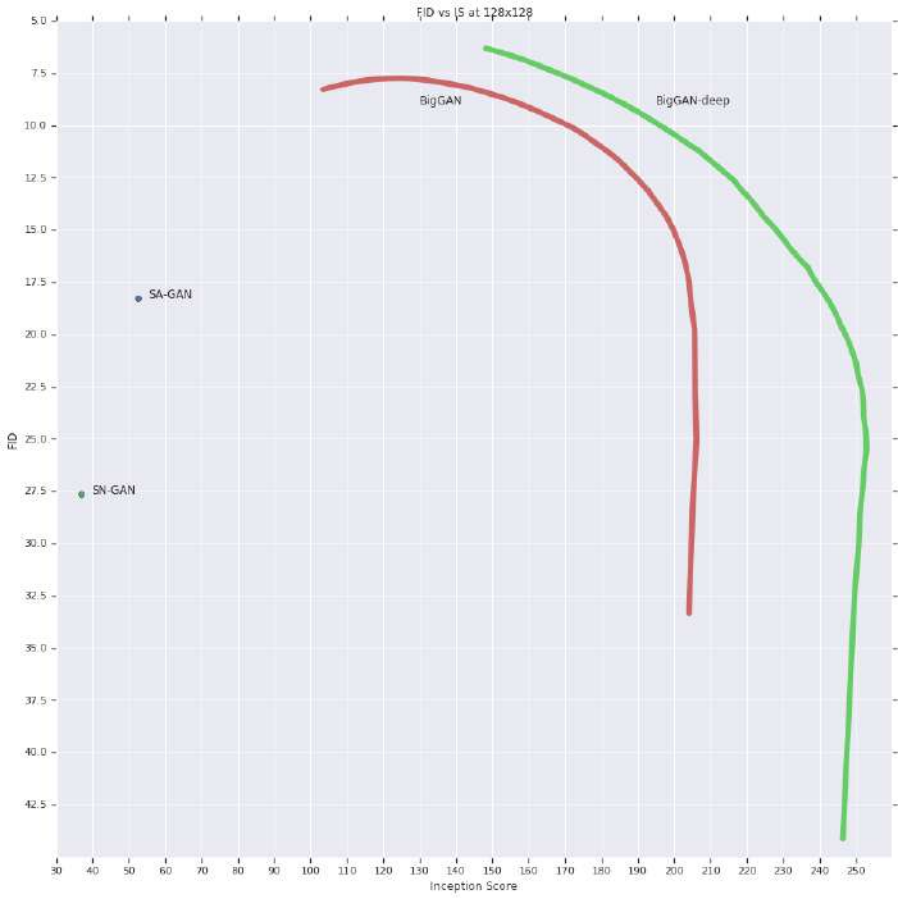


Figure 17: IS vs. FID at 128×128. Scores are averaged across three random seeds.

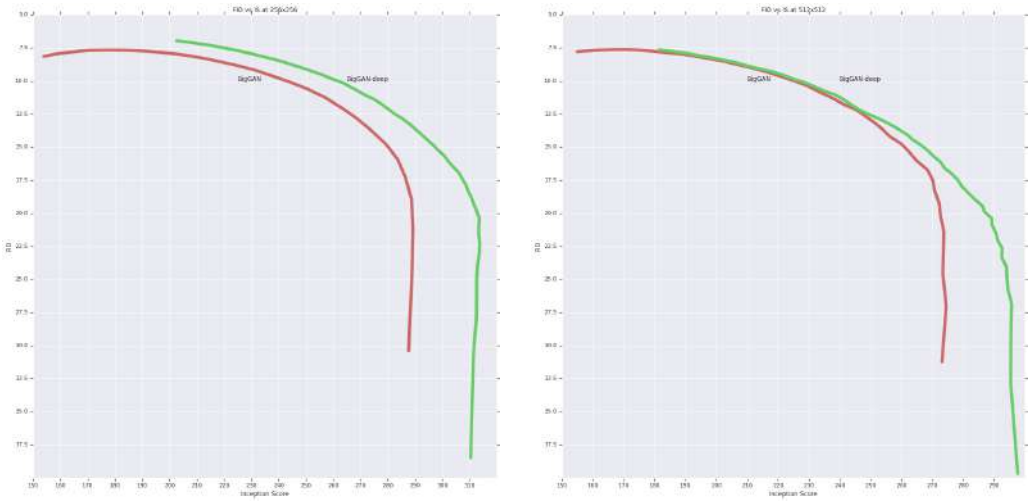


Figure 18: IS vs. FID at 256 and 512 pixels. Scores are averaged across three random seeds for 256.

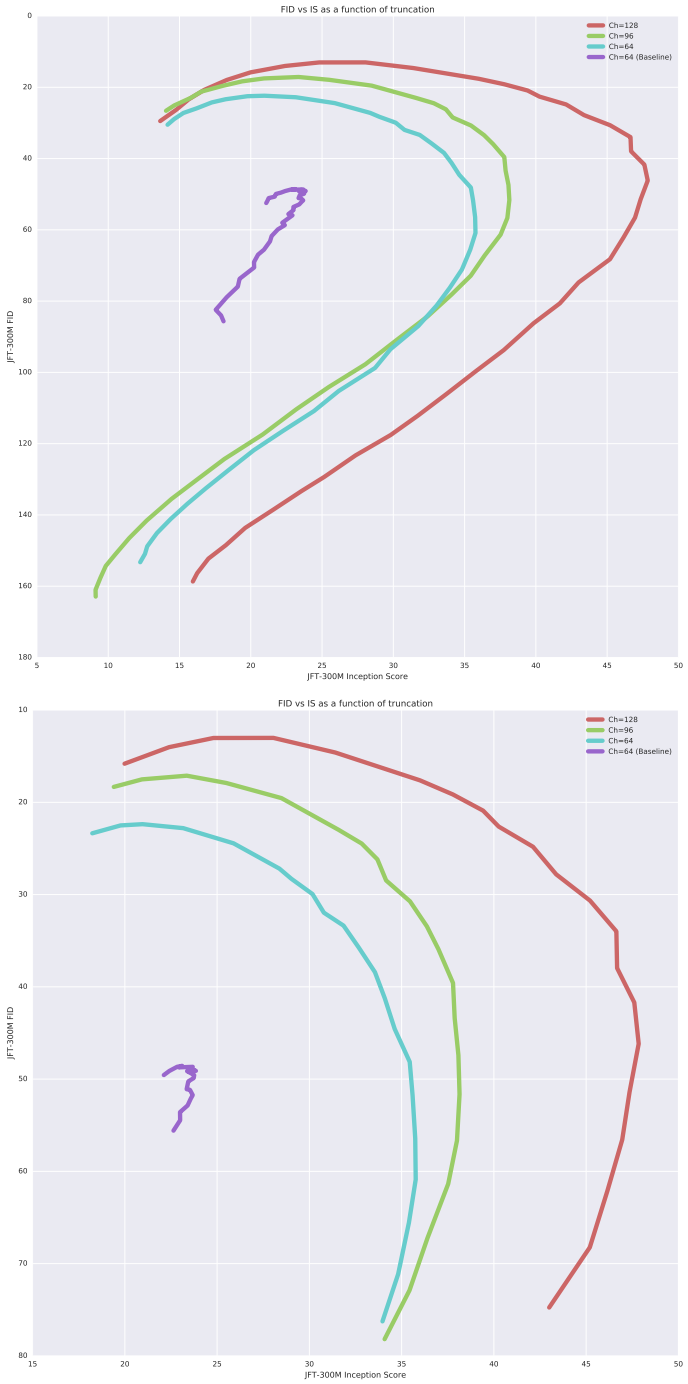


Figure 19: JFT-300M IS vs. FID at 256×256. We show truncation values from $\sigma = 0$ to $\sigma = 2$ (top) and from $\sigma = 0.5$ to $\sigma = 1.5$ (bottom). Each curve corresponds to a row in Table 3. The curve labeled with *baseline* corresponds to the first row (with orthogonal regularization and other techniques disabled), while the rest correspond to rows 2-4 – the same architecture at different capacities (*Ch*).

APPENDIX E CHOOSING LATENT SPACES

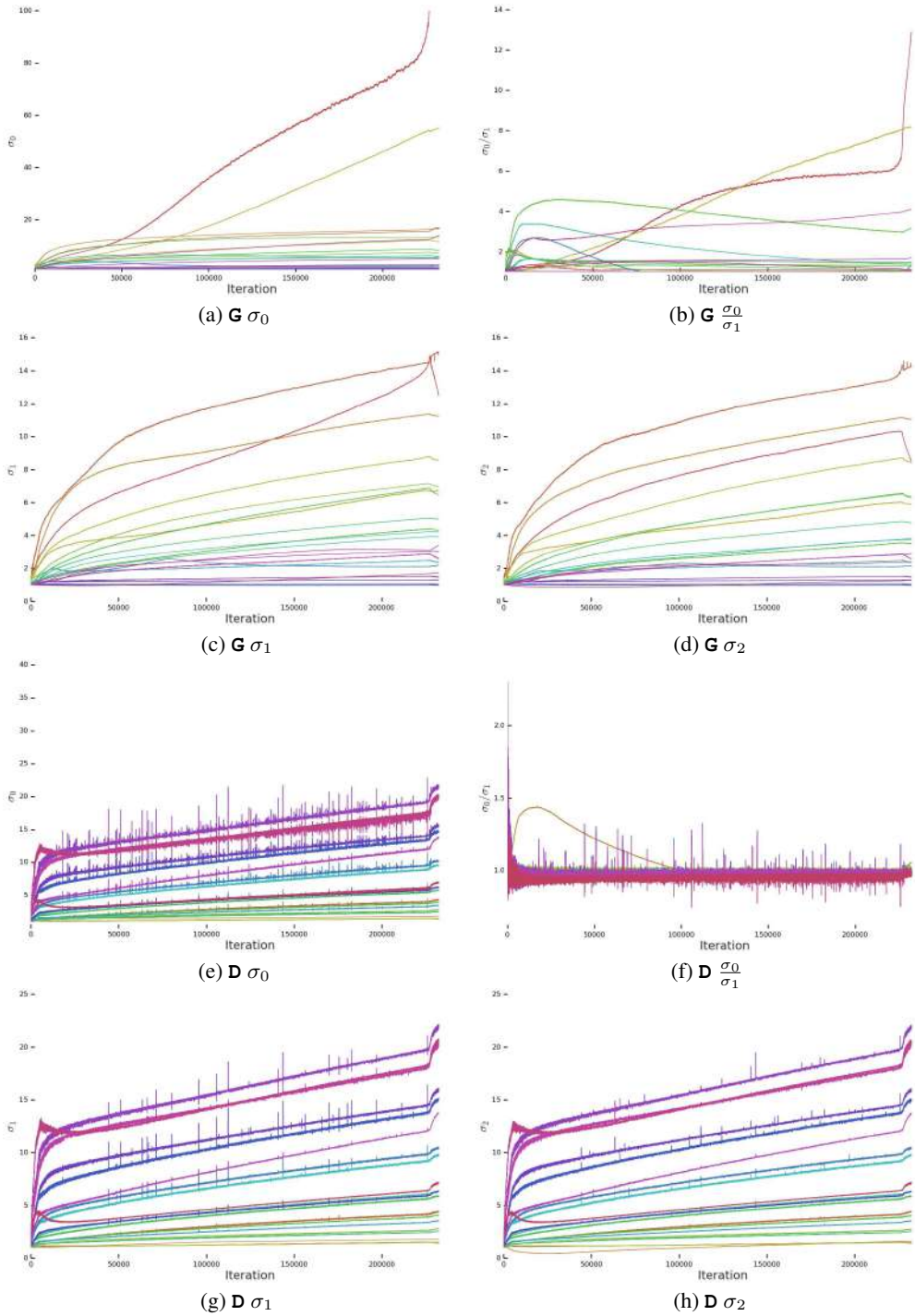
While most previous work has employed $\mathcal{N}(0, I)$ or $\mathcal{U}[-1, 1]$ as the prior for z (the noise input to $\mathbf{G}$), we are free to choose any latent distribution from which we can sample. We explore the choice of latents by considering an array of possible designs, described below. For each latent, we provide the intuition behind its design and briefly describe how it performs when used as a drop-in replacement for $z \sim \mathcal{N}(0, I)$ in an SA-GAN baseline. As the Truncation Trick proved more beneficial than switching to any of these latents, we do not perform a full ablation study, and employ $z \sim \mathcal{N}(0, I)$ for our main results to take full advantage of truncation. The two latents which we find to work best without truncation are Bernoulli $\{0, 1\}$ and Censored Normal $\max(\mathcal{N}(0, I), 0)$, both of which improve speed of training and lightly improve final performance, but are less amenable to truncation.

We also ablate the choice of latent space dimensonality (which by default is $z \in \mathbb{R}^{128}$), finding that we are able to successfully train with latent dimensions as low as $z \in \mathbb{R}^8$, and that with $z \in \mathbb{R}^{32}$ we see a minimal drop in performance. While this is substantially smaller than many previous works, direct comparison to single-class networks (such as those in Karras et al. (2018), which employ a $z \in \mathbb{R}^{512}$ latent space on a highly constrained dataset with 30,000 images) is improper, as our networks have additional class information provided as input.

LATENTS

- $\mathcal{N}(0, I)$. A standard choice of the latent space which we use in the main experiments.
- $\mathcal{U}[-1, 1]$. Another standard choice; we find that it performs similarly to $\mathcal{N}(0, I)$.
- Bernoulli $\{0, 1\}$. A discrete latent might reflect our prior that underlying factors of variation in natural images are not continuous, but discrete (one feature is present, another is not). This latent outperforms $\mathcal{N}(0, I)$ (in terms of IS) by 8% and requires 60% fewer iterations.
- $\max(\mathcal{N}(0, I), 0)$, also called Censored Normal. This latent is designed to introduce sparsity in the latent space (reflecting our prior that certain latent features are sometimes present and sometimes not), but also allow those latents to vary continuously, expressing different degrees of intensity for latents which are active. This latent outperforms $\mathcal{N}(0, I)$ (in terms of IS) by 15-20% and tends to require fewer iterations.
- Bernoulli $\{-1, 1\}$. This latent is designed to be discrete, but not sparse (as the network can learn to activate in response to negative inputs). This latent performs near-identically to $\mathcal{N}(0, I)$.
- Independent Categorical in $\{-1, 0, 1\}$, with equal probability. This distribution is chosen to be discrete and have sparsity, but also to allow latents to take on both positive and negative values. This latent performs near-identically to $\mathcal{N}(0, I)$.
- $\mathcal{N}(0, I)$ multiplied by Bernoulli $\{0, 1\}$. This distribution is chosen to have continuous latent factors which are also sparse (with a peak at zero), similar to Censored Normal but not constrained to be positive. This latent performs near-identically to $\mathcal{N}(0, I)$.
- Concatenating $\mathcal{N}(0, I)$ and Bernoulli $\{0, 1\}$, each taking half of the latent dimensions. This is inspired by Chen et al. (2016), and is chosen to allow some factors of variation to be discrete, while others are continuous. This latent outperforms $\mathcal{N}(0, I)$ by around 5%.
- Variance annealing: we sample from $\mathcal{N}(0, \sigma I)$, where σ is allowed to vary over training. We compared a variety of piecewise schedules and found that starting with $\sigma = 2$ and annealing towards $\sigma = 1$ over the course of training mildly improved performance. The space of possible variance schedules is large, and we did not explore it in depth – we suspect that a more principled or better-tuned schedule could more strongly impact performance.
- Per-sample variable variance: $\mathcal{N}(0, \sigma_i I)$, where $\sigma_i \sim \mathcal{U}[\sigma_l, \sigma_h]$ independently for each sample i in a batch, and (σ_l, σ_h) are hyperparameters. This distribution was chosen to try and improve amenability to the Truncation Trick by feeding the network noise samples with non-constant variance. This did not appear to affect performance, but we did not explore it in depth. One might also consider scheduling (σ_l, σ_h), similar to variance annealing.

APPENDIX F MONITORED TRAINING STATISTICS



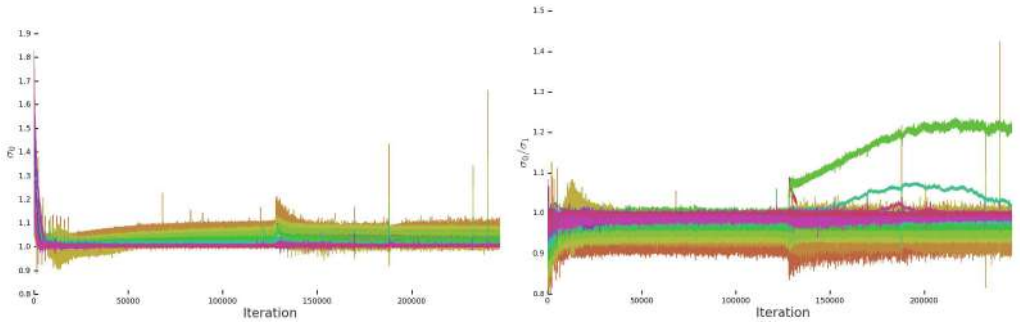
(a) **G** σ_0 (b) **G** $\frac{\sigma_0}{\sigma_1}$

(c) **G** σ_1 (d) **G** σ_2

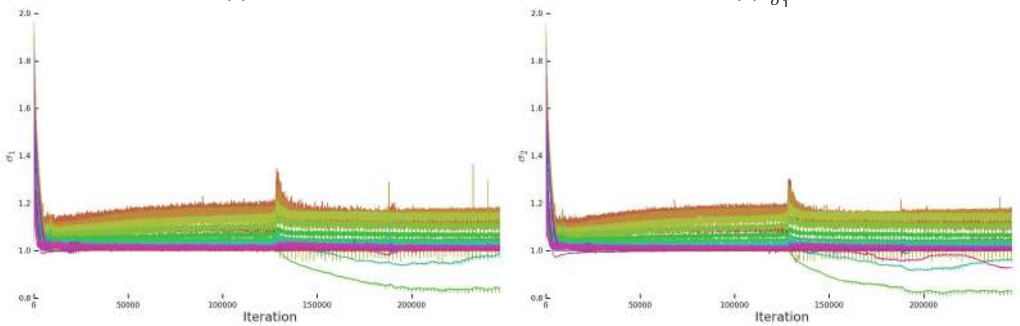
(e) **D** σ_0 (f) **D** $\frac{\sigma_0}{\sigma_1}$

(g) **D** σ_1 (h) **D** σ_2

Figure 20: Training statistics for a typical model without special modifications. Collapse occurs after 200000 iterations.

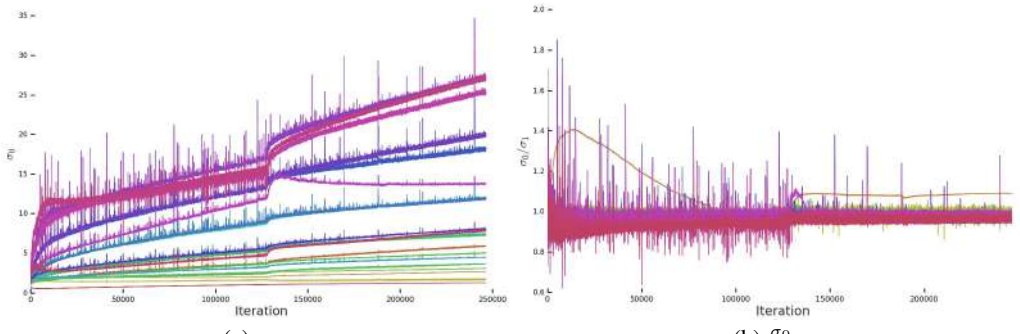


(a) σ_0 (b) $\frac{\sigma_0}{\sigma_1}$

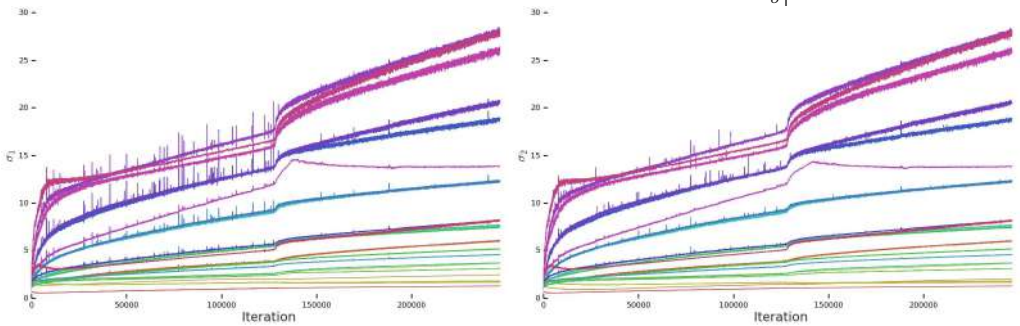


(c) σ_1 (d) σ_2

Figure 21: **G** training statistics with σ_0 in **G** regularized towards 1. Collapse occurs after 125000 iterations.

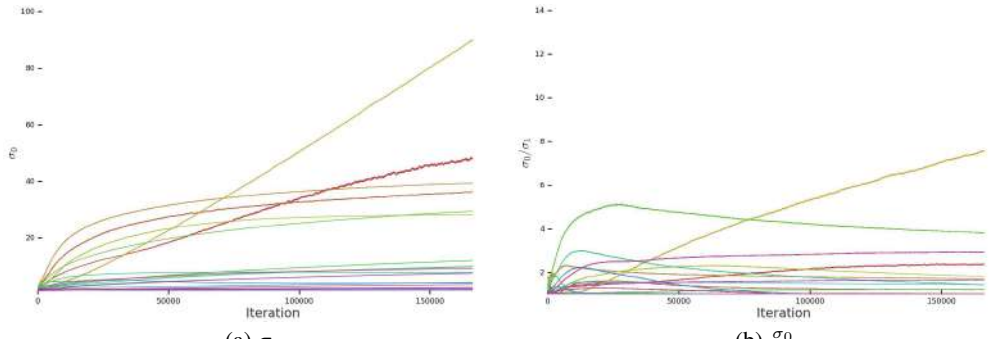


(a) σ_0 (b) $\frac{\sigma_0}{\sigma_1}$

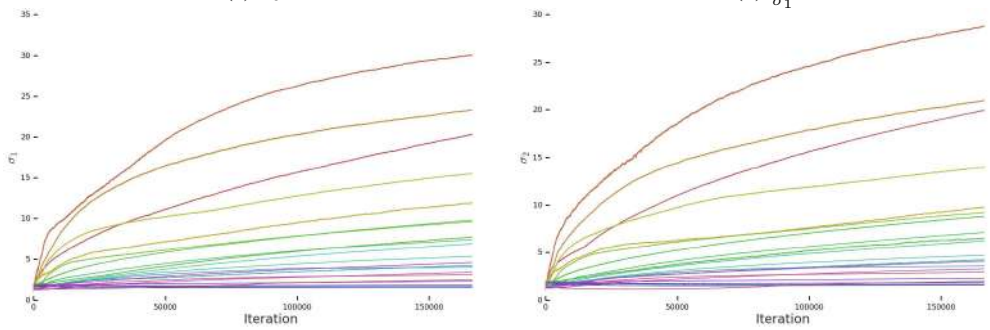


(c) σ_1 (d) σ_2

Figure 22: **D** training statistics with σ_0 in **G** regularized towards 1. Collapse occurs after 125000 iterations.

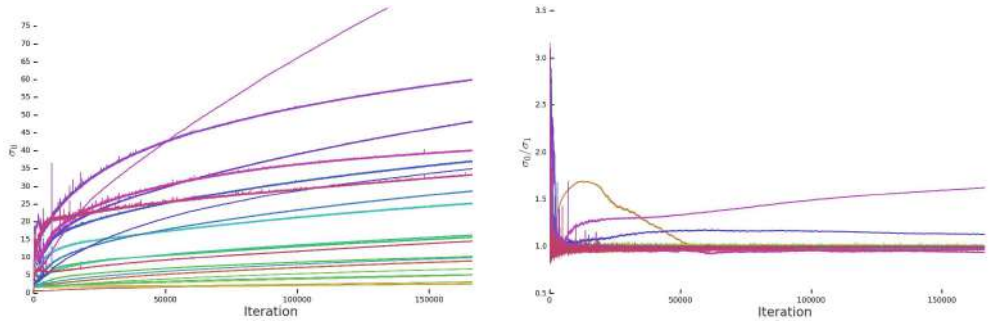


(a) σ_0 (b) $\frac{\sigma_0}{\sigma_1}$

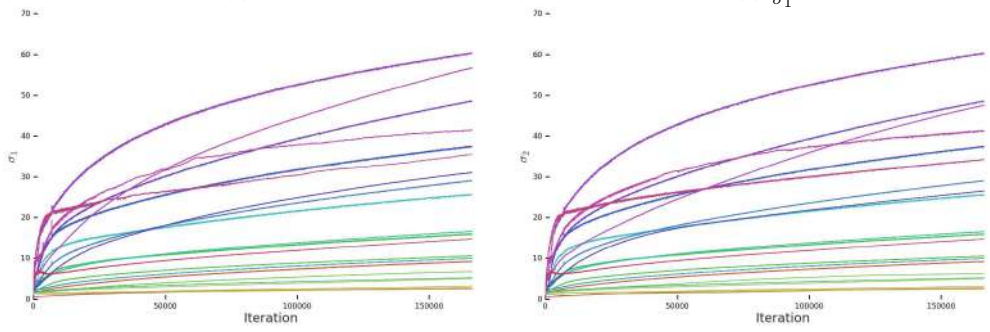


(c) σ_1 (d) σ_2

Figure 23: **G** training statistics with an R1 Gradient Penalty of strength 10 on **D**. This model does not collapse, but only reaches a maximum IS of 55.

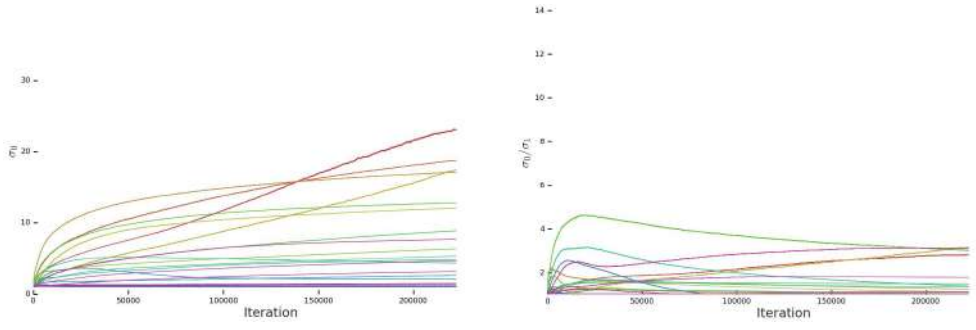


(a) σ_0 (b) $\frac{\sigma_0}{\sigma_1}$

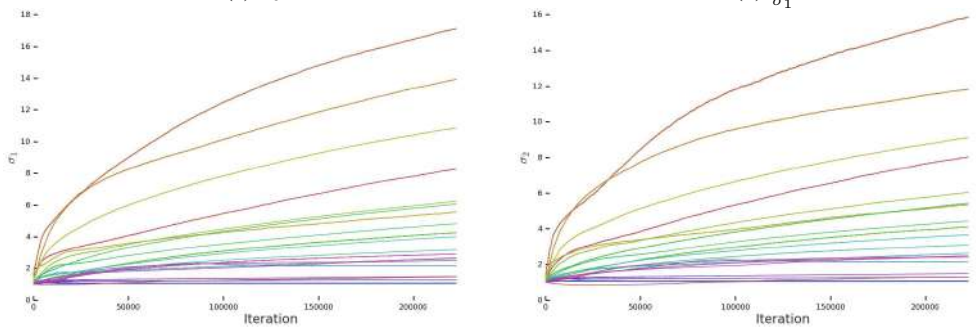


(c) σ_1 (d) σ_2

Figure 24: **D** training statistics with an R1 Gradient Penalty of strength 10 on **D**. This model does not collapse, but only reaches a maximum IS of 55.

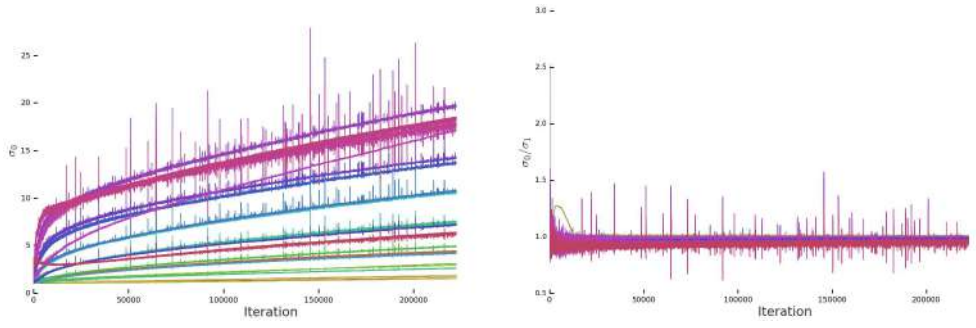


(a) σ_0 (b) $\frac{\sigma_0}{\sigma_1}$

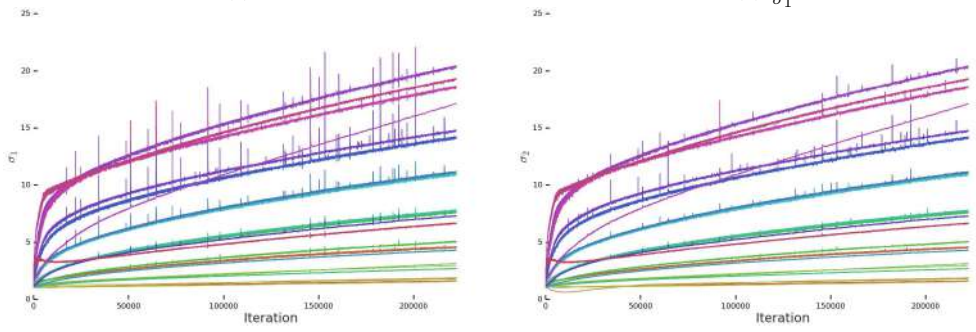


(c) σ_1 (d) σ_2

Figure 25: **G** training statistics with Dropout (keep probability 0.8) applied to the last feature layer of **D**. This model does not collapse, but only reaches a maximum IS of 70.

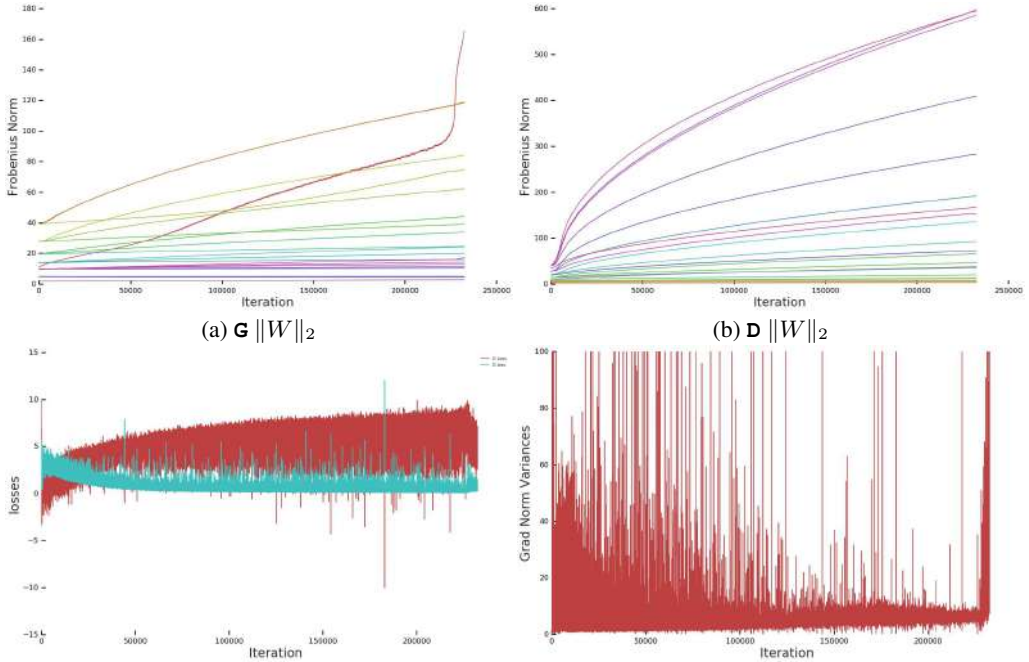


(a) σ_0 (b) $\frac{\sigma_0}{\sigma_1}$



(c) σ_1 (d) σ_2

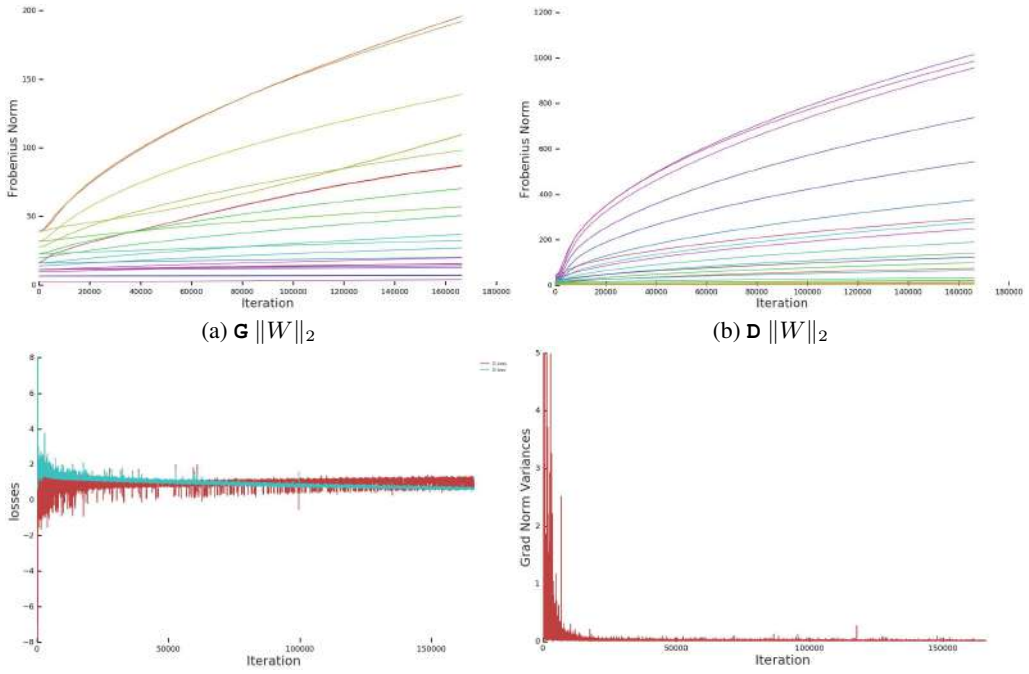
Figure 26: **D** training statistics with Dropout (keep probability 0.8) applied to the last feature layer of **D**. This model does not collapse, but only reaches a maximum IS of 70.



(a) **G** $\|W\|_2$ (b) **D** $\|W\|_2$

(c) losses (d) Variance of all gradient norms in **G** and **D**

Figure 27: Additional training statistics for a typical model without special modifications. Collapse occurs after 200000 iterations.



(a) **G** $\|W\|_2$ (b) **D** $\|W\|_2$

(c) losses (d) Variance of all gradient norms in **G** and **D**

Figure 28: Additional training statistics with an R1 Gradient Penalty of strength 10 on **D**. This model does not collapse, but only reaches a maximum IS of 55.

APPENDIX G ADDITIONAL DISCUSSION: STABILITY AND COLLAPSE

In this section, we present and discuss additional investigations into the stability of our models, expanding upon the discussion in Section 4.

G.1 INTERVENING BEFORE COLLAPSE

The symptoms of collapse are sharp and sudden, with sample quality dropping from its peak to its lowest value over the course of a few hundred iterations. We can detect this collapse when the singular values in **G** explode, but while the (unnormalized) singular values grow throughout training, there is no consistent threshold at which collapse occurs. This raises the question of whether it is possible to prevent or delay collapse by taking a model checkpoint several thousand iterations before collapse, and continuing training with some hyperparameters modified (e.g., the learning rate).

We conducted a range of intervention experiments wherein we took checkpoints of a collapsed model ten or twenty thousand iterations before collapse, changed some aspect of the training setup, then observed whether collapse occurred, when it occurred relative to the original collapse, and the final performance attained at collapse.

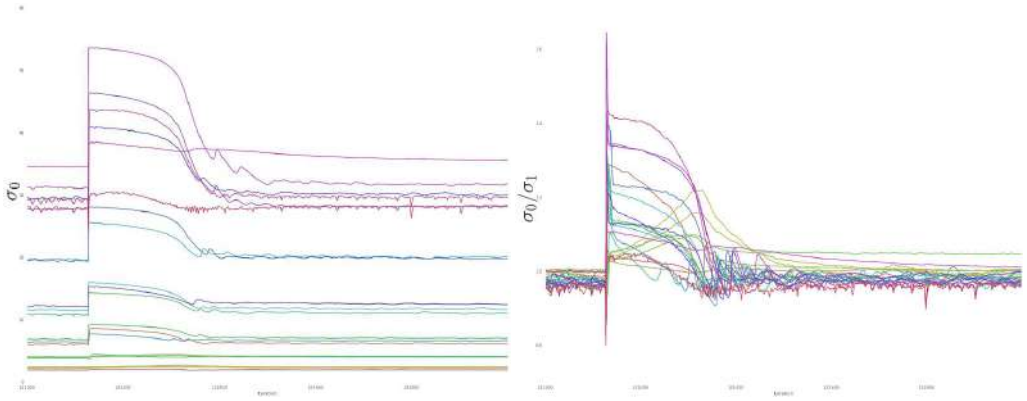
We found that increasing the learning rates (relative to their initial values) in either **G** or **D**, or both **G** and **D**, led to immediate collapse. This occurred even when doubling the learning rates from $2 \cdot 10^{-4}$ in **D** and $5 \cdot 10^{-5}$ in **G**, to $4 \cdot 10^{-4}$ in **D** and $1 \cdot 10^{-4}$ in **G**, a setting which is not normally unstable when used as the initial learning rates. We also tried changing the momentum terms (Adam's β_1 and β_2), or resetting the momentum vectors to zero, but this tended to either make no difference or, when increasing the momentum, cause immediate collapse.

We found that decreasing the learning rate in **G**, but keeping the learning rate in **D** unchanged could delay collapse (in some cases by over one hundred thousand iterations), but also crippled training—once the learning rate in **G** was decayed, performance either stayed constant or slowly decayed. Conversely, reducing the learning rate in **D** while keeping **G**'s learning rate led to immediate collapse. We hypothesize that this is because of the need for **D** to remain optimal throughout training—if its learning rate is reduced, it can no longer "keep up" with **G**, and training collapses. With this in mind, we also tried increasing the number of **D** steps per **G** step, but this either had no effect, or delayed collapse at the cost of crippling training (similar to decaying **G**'s learning rate).

To further illuminate these dynamics, we construct two additional intervention experiments, one where we freeze **G** before collapse (by ceasing all parameter updates) and observe whether **D** remains stable, and the reverse, where we freeze **D** before collapse and observe whether **G** remains stable. We find that when **G** is frozen, **D** remains stable, and slowly reduces both components of its loss towards zero. However, when **D** is frozen, **G** immediately and dramatically collapses, maxing out **D**'s loss to values upwards of 300, compared to the normal range of 0 to 3.

This leads to two conclusions: first, as has been noted in previous works (Miyato et al., 2018; Gulrajani et al., 2017; Zhang et al., 2018), **D** must remain optimal with respect to **G** both for stability and to provide useful gradient information. The consequence of **G** being allowed to win the game is a complete breakdown of the training process, regardless of **G**'s conditioning or optimization settings. Second, favoring **D** over **G** (either by training it with a larger learning rate, or for more steps) is insufficient to ensure stability even if **D** is well-conditioned. This suggests either that in practice, an optimal **D** is necessary but insufficient for training stability, or that some aspect of the system results in **D** not being trained towards optimality. With the latter possibility in mind, we take a closer look at the noise in **D**'s spectra in the following section.

G.2 SPIKES IN THE DISCRIMINATOR'S SPECTRA



(a) **D** σ_0 (b) **D** $\frac{\sigma_0}{\sigma_1}$

Figure 29: A closeup of **D**'s spectra at a noise spike.

If some element of **D**'s training process results in undesirable dynamics, it follows that the behavior of **D**'s spectra may hold clues as to what that element is. The top three singular values of **D** differ from **G**'s in that they have a large noise component, tend to grow throughout training but only show a small response to collapse, and the ratio of the first two singular values tends to be centered around one, suggesting that the spectra of **D** have a slow decay. When viewed up close (Figure 29), the noise spikes resemble an impulse response: at each spike, the spectra jump upwards, then slowly decrease, with some oscillation.

One possible explanation is that this behavior is a consequence of **D** memorizing the training data, as suggested by experiments in Section 4.2. As it approaches perfect memorization, it receives less and less signal from real data, as both the original GAN loss and the hinge loss provide zero gradients when **D** outputs a confident and correct prediction for a given example. If the gradient signal from real data attenuates to zero, this can result in **D** eventually becoming biased due to exclusively received gradients that encourage its outputs to be negative. If this bias passes a certain threshold, **D** will eventually misclassify a large number of real examples and receive a large gradient encouraging positive outputs, resulting in the observed impulse responses.

This argument suggests several fixes. First, one might consider an unbounded loss (such as the Wasserstein loss (Arjovsky et al., 2017)) which would not suffer this gradient attentuation. We found that even with gradient penalties and brief re-tuning of optimizer hyperparameters, our models did not stably train for more than a few thousand iterations with this loss. We instead explored changing the margin of the hinge loss as a partial compromise: for a given model and minibatch of data, increasing the margin will result in more examples falling within the margin, and thus contributing to the loss.[3]. Training with a smaller margin (by a factor of 2) measurably reduces performance, but training with a larger margin (by up to a factor of 3) does not prevent collapse or reduce the noise in **D**'s spectra. Increasing the margin beyond 3 results in unstable training similar to using the Wasserstein loss. Finally, the memorization argument might suggest that using a smaller **D** or using dropout in **D** would improve training by reducing its capacity to memorize, but in practice this degrades training.

[3]Unconstrained models could easily learn a different output scale to account for this margin, but the use of Spectral Normalization constrains our models and makes the specific selection of the margin meaningful.

APPENDIX H NEGATIVE RESULTS

We explored a range of novel and existing techniques which ended up degrading or otherwise not affecting performance in our setting. We report them here; our evaluations for this section are not as thorough as those for the main architectural choices.

Our intention in reporting these results is to save time for future work, and to give a more complete picture of our attempts to improve performance or stability. We note, however, that these results must be understood to be specific to the particular setup we used. A pitfall of reporting negative results is that one might report that a particular technique doesn't work, when the reality is that this technique did not have the desired effect when applied in a particular way to a particular problem. Drawing overly general conclusions might close off potentially fruitful avenues of research.

- We found that doubling the depth (by inserting an additional Residual block after every up- or down-sampling block) hampered performance.
- We experimented with sharing class embeddings between both **G** and **D** (as opposed to just within **G**). This is accomplished by replacing **D**'s class embedding with a projection from **G**'s embeddings, as is done in **G**'s BatchNorm layers. In our initial experiments this seemed to help and accelerate training, but we found this trick scaled poorly and was sensitive to optimization hyperparameters, particularly the choice of number of **D** steps per **G** step.
- We tried replacing BatchNorm in **G** with WeightNorm (Salimans & Kingma, 2016), but this crippled training. We also tried removing BatchNorm and only having Spectral Normalization, but this also crippled training.
- We tried adding BatchNorm to **D** (both class-conditional and unconditional) in addition to Spectral Normalization, but this crippled training.
- We tried varying the choice of location of the attention block in **G** and **D** (and inserting multiple attention blocks at different resolutions) but found that at 128×128 there was no noticeable benefit to doing so, and compute and memory costs increased substantially. We found a benefit to moving the attention block up one stage when moving to 256×256, which is in line with our expectations given the increased resolution.
- We tried using filter sizes of 5 or 7 instead of 3 in either **G** or **D** or both. We found that having a filter size of 5 in **G** only provided a small improvement over the baseline but came at an unjustifiable compute cost. All other settings degraded performance.
- We tried varying the dilation for convolutional filters in both **G** and **D** at 128×128, but found that even a small amount of dilation in either network degraded performance.
- We tried bilinear upsampling in **G** in place of nearest-neighbors upsampling, but this degraded performance.
- In some of our models, we observed class-conditional mode collapse, where the model would only output one or two samples for a subset of classes but was still able to generate samples for all other classes. We noticed that the collapsed classes had embedings which had become very large relative to the other embeddings, and attempted to ameliorate this issue by applying weight decay to the shared embedding only. We found that small amounts of weight decay (10^{-6}) instead degraded performance, and that only even smaller values (10^{-8}) did not degrade performance, but these values were also too small to prevent the class vectors from exploding. Higher-resolution models appear to be more resilient to this problem, and none of our final models appear to suffer from this type of collapse.
- We experimented with using MLPs instead of linear projections from **G**'s class embeddings to its BatchNorm gains and biases, but did not find any benefit to doing so. We also experimented with Spectrally Normalizing these MLPs, and with providing these (and the linear projections) with a bias at their output, but did not notice any benefit.
- We tried gradient norm clipping (both the global variant typically used in recurrent networks, and a local version where the clipping value is determined on a per-parameter basis) but found this did not alleviate instability.

APPENDIX I HYPERPARAMETERS

We performed various hyperparameter sweeps in this work:

- We swept the Cartesian product of the learning rates for each network through [10^{-5}, $5 \cdot 10^{-5}$, 10^{-4}, $2 \cdot 10^{-4}$, $4 \cdot 10^{-4}$, $8 \cdot 10^{-4}$, 10^{-3}], and initially found that the SA-GAN settings (**G**'s learning rate 10^{-4}, **D**'s learning rate $4 \cdot 10^{-4}$) were optimal at lower batch sizes; we did not repeat this sweep at higher batch sizes but did try halving and doubling the learning rate, arriving at the halved settings used for our experiments.
- We swept the R1 gradient penalty strength through [10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1, 2, 3, 5, 10]. We find that the strength of the penalty correlates negatively with performance, but that settings above 0.5 impart training stability.
- We swept the keep probabilities for DropOut in the final layer of **D** through [0.5, 0.6, 0.7, 0.8, 0.9, 0.95]. We find that DropOut has a similar stabilizing effect to R1 but also degrades performance.
- We swept **D**'s Adam β_1 parameter through [0.1, 0.2, 0.3, 0.4, 0.5] and found it to have a light regularization effect similar to DropOut, but not to significantly improve results. Higher β_1 terms in either network crippled training.
- We swept the strength of the modified Orthogonal Regularization penalty in **G** through [10^{-5}, $5 \cdot 10^{-5}$, 10^{-4}, $5 \cdot 10^{-4}$, 10^{-3}, 10^{-2}], and selected 10^{-4}.