Generating Images with Perceptual Similarity Metrics based on Deep Networks

Alexey Dosovitskiy DOSOVITS@CS.UNI-FREIBURG.DE
University of Freiburg, Germany

Thomas Brox BROX@CS.UNI-FREIBURG.DE
University of Freiburg, Germany

Abstract

Image-generating machine learning models are typically trained with loss functions based on distance in the image space. This often leads to over-smoothed results. We propose a class of loss functions, which we call deep perceptual similarity metrics (DeePSiM), that mitigate this problem. Instead of computing distances in the image space, we compute distances between image features extracted by deep neural networks. This metric better reflects perceptually similarity of images and thus leads to better results. We show three applications: autoencoder training, a modification of a variational autoencoder, and inversion of deep convolutional networks. In all cases, the generated images look sharp and resemble natural images.

1. Introduction

Recently there has been a surge of interest in training neural networks to generate images. These are being used for a wide variety of applications: unsupervised and semi-supervised learning, generative models, analysis of learned representations, analysis by synthesis, learning of 3D representations, future prediction in videos. Nevertheless, there is little work on studying loss functions which are appropriate for the image generation task. Typically used squared Euclidean distance between images often yields blurry results, see Fig.1b. This is especially the case when there is inherent uncertainty in the prediction. For example, suppose we aim to reconstruct an image from its feature representation. The precise location of all details may not be preserved in the features. A loss in image space leads to averaging all likely locations of details, and hence the reconstruction looks blurry.

However, exact locations of all fine details are not important for perceptual similarity of images. But the distribution of these details plays a key role. Our main insight is that invariance to irrelevant transformations and sensitivity to local image statistics can be achieved by measuring distances in a suitable feature space. In fact, convolutional networks provide a feature representation with desirable properties. They are invariant to small smooth deformations, but sensitive to perceptually important image properties, for example sharp edges and textures.

Using a distance in feature space alone, however, does not yet yield a good loss function; see Fig. 1d. Since feature representations are typically contractive, many images, including non-natural ones, get mapped to the same feature vector. Hence, we must introduce a natural image prior. To this end, we build upon adversarial training as proposed by Goodfellow et al. (2014). We train a discriminator network to distinguish the output of the generator from real images. The objective of the generator is to trick the discriminator, i.e., to generate images that the discriminator cannot distinguish from real ones. This yields a natural image prior that selects from all potential generator outputs the most realistic one. A combination of similarity in an appropriate feature space with adversarial training allows to obtain the best results; see Fig. 1e.

We show three example applications: image compression with an autoencoder, a generative model based on a variational autoencoder, and inversion of the AlexNet convolutional network. We demonstrate that an autoencoder with

Figure 1: Reconstructions from layer FC6 of AlexNet with different losses.

DeePSiM loss can compress images while preserving information about fine structures. On the generative modeling side, we show that a version of a variational autoencoder trained with the new loss produces images with realistic image statistics. Finally, reconstructions obtained with our method from high-level activations of AlexNet are dramatically better than with existing approaches. They demonstrate that even the predicted class probabilities contain rich texture, color, and position information.

2. Related work

There is a long history of neural network based models for image generation. A prominent class of probabilistic models of images are restricted Boltzmann machines (Hinton & Sejnowski, 1986; Smolensky, 1987; Hinton & Salakhutdinov, 2006) and their deep variants (Hinton et al., 2006; Salakhutdinov & Hinton, 2009; Lee et al., 2009). Autoencoders (Hinton & Salakhutdinov, 2006; Vincent et al., 2008) have been widely used for unsupervised learning and generative modeling, too. Recently, stochastic neural networks (Bengio et al., 2014; Kingma et al., 2014; Gregor et al., 2015) have become popular, and deterministic networks are being used for image generation tasks (Dosovitskiy et al., 2015b). In all these models, loss is measured in the image space. By combining convolutions and unpooling (upsampling) layers (Lee et al., 2009; Goodfellow et al., 2014; Dosovitskiy et al., 2015b) these models can be applied to large images.

There is a large body of work on assessing the perceptual similarity of images. Some prominent examples are the visible differences predictor (Daly, 1993), the spatio-temporal model for moving picture quality assessment (van den Branden Lambrecht & Verscheure, 1996), and the perceptual distortion metric of Winkler (1998). The most popular perceptual image similarity metric is the structural similarity metric (SSIM) (Wang et al., 2004), which compares the local statistics of image patches. We are not aware of any work making use of similarity metrics for machine learning, except a recent pre-print of Ridgeway et al. (2015). They train autoencoders by directly maximizing the SSIM similarity of images. This resembles in spirit what we do, but technically is very different. While psychophysical experiments go out of scope of this paper, we believe that deep learned feature representations have better potential than shallow hand-designed SSIM.

Generative adversarial networks (GANs) have been proposed by Goodfellow et al. (2014). In theory, this training procedure can lead to a generator that perfectly models the data distribution. Practically, training GANs is difficult and often leads to oscillatory behavior, divergence, or modeling only part of the data distribution. Recently, several modifications have been proposed that make GAN training more stable. Denton et al. (2015) employ a multi-scale approach, gradually generating higher resolution images. Radford et al. (2015) make use of a convolutional-deconvolutional architecture and batch normalization.

GANs can be trained conditionally by feeding the conditioning variable to both the discriminator and the generator (Mirza & Osindero, 2014). Usually this conditioning variable is a one-hot encoding of the object class in the input image. Such GANs learn to generate images of objects from a given class. Recently Mathieu et al. (2015) used GANs for predicting future frames in videos by conditioning on previous frames. Our approach looks similar to a conditional GAN. However, in a GAN there is no loss directly comparing the generated image to some ground truth. We found that the feature loss introduced in the present paper is essential to train on complicated tasks such as feature inversion.

Most related is concurrent work of Larsen et al. (2015). The general idea is the same — to measure the similarity not in the image space, but rather in a feature space. They also use adversarial training to improve the realism of the generated images. However, Larsen et al. (2015) only apply this approach to a variational autoencoder trained on images of faces, and measure the similarity between features extracted from the discriminator. Our approach is much more general, we apply it to various natural images, and we demonstrate three different applications.

3. Model

Suppose we are given a supervised learning task and a training set of input-target pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$, $\mathbf{x}_i \in \mathbb{R}^I$, $\mathbf{y}_i \in \mathbb{R}^{W \times H \times C}$. Inputs and outputs can be arbitrary vectors. In this work, we focus on targets that are images with an arbitrary number of channels.

The aim is to learn the parameters θ of a differentiable generator function $G_\theta(\cdot)\colon \mathbb{R}^I \to \mathbb{R}^{W \times H \times C}$ that optimally approximates the input-target dependency according to a loss function $\mathcal{L}(G_\theta(\mathbf{x}), \mathbf{y})$. Typical choices are squared Euclidean (SE) loss $\mathcal{L}_2(G_\theta(\mathbf{x}), \mathbf{y}) = ||G_\theta(\mathbf{x}) - \mathbf{y}||_2^2$ or ℓ_1 loss $\mathcal{L}_1(G_\theta(\mathbf{x}), \mathbf{y}) = ||G_\theta(\mathbf{x}) - \mathbf{y}||_1$. As we demonstrate in this paper, these losses are suboptimal for some image generation tasks.

We propose a new class of losses, which we call DeePSiM. These go beyond simple distances in image space and can capture complex and perceptually important properties of images. These losses are weighted sums of three terms: feature loss $\mathcal{L}_{feat}$, adversarial loss $\mathcal{L}_{adv}$, and pixel space loss $\mathcal{L}_{img}$:

$$\mathcal{L} = \lambda_{feat}\,\mathcal{L}_{feat} + \lambda_{adv}\,\mathcal{L}_{adv} + \lambda_{img}\,\mathcal{L}_{img}. \qquad (1)$$

They correspond to a network architecture, an overview of

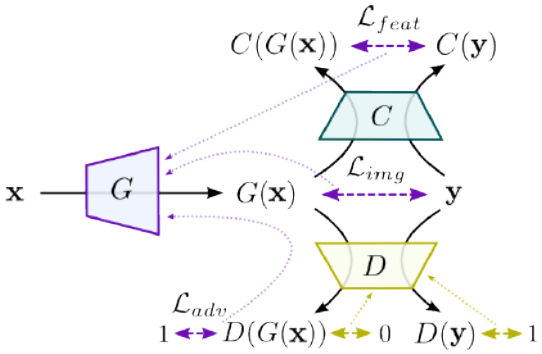

Figure 2: Schematic of our model. Black solid lines denote the forward pass. Dashed lines with arrows on both ends are the losses. Thin dashed lines denote the flow of gradients.

which is shown in Fig. 2. The architecture consists of three convolutional networks: the generator G that implements the generator function, the discriminator D_φ that discriminates generated images from natural images, and the comparator C that computes features from images.

Loss in feature space. Given a differentiable comparator $C\colon \mathbb{R}^{W\times H\times C} \to \mathbb{R}^F$, we define

$$\mathcal{L}_{feat} = \sum_i ||C(G_\theta(\mathbf{x}_i)) - C(\mathbf{y}_i)||_2^2. \quad (2)$$

C may be fixed or may be trained; for example, it can be a part of the generator or the discriminator.

$\mathcal{L}_{feat}$ alone does not provide a good loss for training. It is known (Mahendran & Vedaldi, 2015) that optimizing just for similarity in the feature space typically leads to high-frequency artifacts. This is because for each natural image there are many non-natural images mapped to the same feature vector [1]. Therefore, a natural image prior is necessary to constrain the generated images to the manifold of natural images.

Adversarial loss. Instead of manually designing a prior, as in Mahendran & Vedaldi (2015), we learn it with an approach similar to Generative Adversarial Networks (GANs) of Goodfellow et al. (2014). Namely, we introduce a discriminator D_φ which aims to discriminate the generated images from real ones, and which is trained concurrently with the generator G_θ. The generator is trained to "trick" the discriminator network into classifying the generated images as real. Formally, the parameters φ of the discriminator are trained by minimizing

$$\mathcal{L}_{discr} = -\sum_i \log(D_\varphi(\mathbf{y}_i)) + \log(1 - D_\varphi(G_\theta(\mathbf{x}_i))), \quad (3)$$

[1] This is unless the feature representation is specifically designed to map natural and non-natural images far apart, such as the one extracted from the discriminator of a GAN.

and the generator is trained to minimize

$$\mathcal{L}_{adv} = -\sum_i \log D_\varphi(G_\theta(\mathbf{x}_i)). \quad (4)$$

Loss in image space. Adversarial training is known to be unstable and sensitive to hyperparameters. We found that adding a loss in the image space

$$\mathcal{L}_{img} = \sum_i ||G_\theta(\mathbf{x}_i) - \mathbf{y}_i||_2^2. \quad (5)$$

stabilizes training.

3.1. Architectures

Generators. We used several different generators in experiments. They are task-specific, so we describe these in corresponding sections below. All tested generators make use of up-convolutional ('deconvolutional') layers, as in Dosovitskiy et al. (2015b). An up-convolutional layer consists of up-sampling and a subsequent convolution. In this paper we always up-sample by a factor of 2 and a 'bed of nails' upsampling.

In all networks we use leaky ReLU nonlinearities, that is, $LReLU(x) = \max(x, 0) + \alpha \min(x, 0)$. We used $\alpha = 0.3$ in our experiments. All generators have linear output layers.

Comparators. We experimented with four comparators:

1. AlexNet (Krizhevsky et al., 2012) is a network with 5 convolutional and 2 fully connected layers trained on image classification.

2. The network of Wang & Gupta (2015) has the same architecture as AlexNet, but is trained using videos with triplet loss, which enforces frames of one video to be close in the feature space and frames from different videos to be far apart. We refer to this network as VideoNet.

3. AlexNet with random weights.

4. Exemplar-CNN (Dosovitskiy et al., 2015a) is a network with 3 convolutional layers and 1 fully connected layer trained on a surrogate task of discriminating between different image patches.

The exact layers used for comparison are specified in the experiments sections.

Discriminator. The architecture of the discriminator was nearly the same in all experiments. The version used for the autoencoder experiments is shown in Table 1. The discriminator must ensure the local statistics of images to be natural. Therefore after five convolutional layers with occasional stride we perform global average pooling. The result is processed by two fully connected layers, followed by a

Type	conv	conv	conv	conv	conv	pool	fc	fc
InSize	64	29	25	12	10	4	–	–
OutCh	32	64	128	256	256	256	512	2
Kernel	7	5	3	3	3	4	–	–
Stride	2	1	2	1	2	4	–	–

Table 1: Discriminator architecture.

InSize	64	32	32	16	16	8	8	8
OutCh	32	32	64	64	128	128	64	8
Kernel	5	3	5	3	3	3	3	3
Stride	$\downarrow 2$	1	$\downarrow 2$	1	$\downarrow 2$	1	1	1

InSize	8	8	8	16	16	32	32	64
OutCh	64	128	64	64	32	32	16	3
Kernel	3	3	4	3	4	3	4	3
Stride	1	1	$\uparrow 2$	1	$\uparrow 2$	1	$\uparrow 2$	1

Table 2: Autoencoder architecture. **Top:** encoder, **bottom:** decoder. All layers are convolutional or 'up-convolutional'.

2-way softmax. We perform 50% dropout after the global average pooling layer and the first fully connected layer.

There are two modifications to this basic architecture. First, when dealing with large ImageNet (Deng et al., 2009) images we increase the stride in the first layer from 2 to 4. Second, when training networks to invert AlexNet, we additionally feed the features to the discriminator. We process them with two fully connected layers with 1024 and 512 units, respectively. Then we concatenate the result with the output of global average pooling.

3.2. Training details

We modified the *caffe* (Jia et al., 2014) framework to train the networks. For optimization we used Adam (Kingma & Ba, 2015) with momentum $\beta_1 = 0.9$, $\beta_2 = 0.999$ and initial learning rate 0.0002. To prevent the discriminator from overfitting during adversarial training we temporarily stopped updating it if the ratio of $\mathcal{L}_{discr}$ and $\mathcal{L}_{adv}$ was below a certain threshold (0.1 in most experiments). We used batch size 64 in all experiments. We trained for $500,000$-$1,000,000$ mini-batch iterations.

4. Experiments

We started with a simple proof-of-concept experiment showing how DeePSiM can be applied to training autoencoders. Then we used the proposed loss function within the variational autoencoder (VAE) framework. Finally, we applied the method to invert the representation learned by AlexNet and analyzed some properties of the method.

In quantitative comparisons we report normalized Euclidean error $||a - b||_2/N$. The normalization coefficient N is the average of Euclidean distances between all pairs of different samples from the test set. Therefore, the error of 100% means that the algorithm performs the same as randomly drawing a sample from the test set.

4.1. Autoencoder

Here the target of the generator coincides with its input (that is, $\mathbf{y} = \mathbf{x}$), and the task of the generator is to encode the input to a compressed hidden representation and then decode back the image. The architecture is shown in Table 2. All layers are convolutional or up-convolutional.

SE loss	ℓ_1 loss	Our-ExCNN	Our-AlexNet
15.3	15.7	19.8	21.5

Table 3: Normalized Euclidean reconstruction error (in %) of autoencoders trained with different loss functions.

The hidden representation is an 8-channel feature map 8 times smaller than the input image. We trained on the STL-10 (Coates et al., 2011) unlabeled dataset which contains $100,000$ images 96×96 pixels. To prevent overfitting we augmented the data by cropping random 64×64 patches during training.

We experimented with four loss functions: SE and ℓ_1 in the image space, as well as DeePSiM with AlexNet CONV3 or Exemplar-CNN CONV3 as comparator.

Qualitative results are shown in Fig. 3, quantitative results in Table 3. While underperforming in terms of Euclidean loss, our approach can preserve more texture details, resulting in naturally looking non-blurry reconstructions. Interestingly, AlexNet as comparator tends to corrupt fine details (petals of the flower, sails of the ship), perhaps because it has stride of 4 in the first layer. Exemplar-CNN as comparator does not preserve the exact color because it is explicitly trained to be invariant to color changes. We believe that with carefully selected or specifically trained comparators yet better results can be obtained.

We stress that lower Euclidean error does not mean better reconstruction. For example, imagine a black-and-white striped "zebra" pattern. A monotonous gray image will have twice smaller Euclidean error than the same pattern shifted by one stripe width.

Classification. Reconstruction-based models are commonly used for unsupervised feature learning. We checked

SE loss	ℓ_1 loss	Our-ExCNN	Our-AlexNet
34.6 ± 0.6	35.7 ± 0.4	50.1 ± 0.5	52.3 ± 0.6

Table 4: Classification accuracy (in %) on STL with autoencoder features learned with different loss functions.

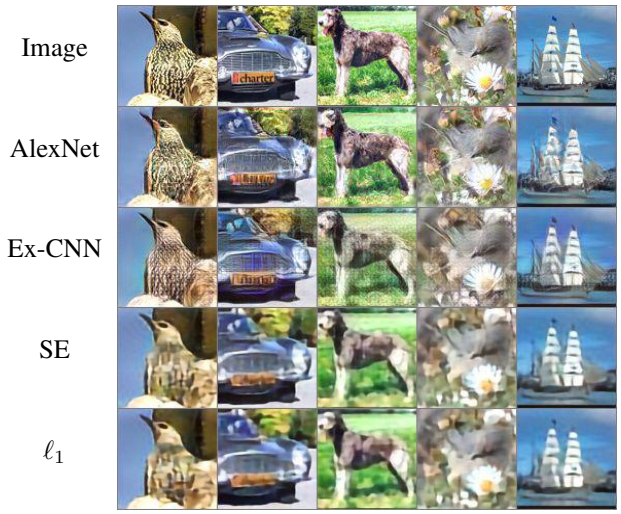

Figure 3: Autoencoder qualitative results. Best viewed on screen.

if our loss functions lead to learning more meaningful representations than usual ℓ_1 and SE losses. To this end, we trained linear SVMs on the 8-channel hidden representations extracted by autoencoders trained with different losses. We are just interested in relative performance and, thus, do not compare to the state of the art. We trained on 10 folds of the STL-10 training set and tested on the test set.

The results are shown in Table 4. As expected, the features learned with DeePSiM perform significantly better, indicating that they contain more semantically meaningful information. This suggests that other losses than standard ℓ_1 and SE may be useful for unsupervised learning. Note that the Exemplar-CNN comparator is trained in an unsupervised way.

4.2. Variational autoencoder

A standard VAE consists of an encoder Enc and a decoder Dec. The encoder maps an input sample x to a distribution over latent variables $z \sim Enc(x) = q(z|x)$. Dec maps from this latent space to a distribution over images $\tilde{x} \sim Dec(z) = p(x|z)$. The loss function is

$$\sum_i -\mathbb{E}_{q(z|x_i)} \log p(x_i|z) + D_{KL}(q(z|x_i)||p(z)), \quad (6)$$

where $p(z)$ is a prior distribution of latent variables and D_{KL} is the Kullback-Leibler divergence. The first term in Eq. 6 is a reconstruction error. If we assume that the decoder predicts a Gaussian distribution at each pixel, then it reduces to squared Euclidean error in the image space. The second term pulls the distribution of latent variables towards the prior. Both $q(z|x)$ and $p(z)$ are commonly assumed to be Gaussian, in which case the KL divergence can be computed analytically. Please refer to Kingma et al. (2014) for details.

We use the proposed loss instead of the first term in Eq. 6. This is similar to Larsen et al. (2015), but the comparator does not have to be a part of the discriminator. Technically, there is little difference from training an autoencoder. First, instead of predicting a single latent vector z we predict two vectors μ and σ and sample $z = \mu + \sigma \odot \varepsilon$, where ε is standard Gaussian (zero mean, unit variance) and $\odot$ is element-wise multiplication. Second, we add the KL divergence term to the loss:

$$\mathcal{L}_{KL} = \frac{1}{2} \sum_i \left(||\mu_i||_2^2 + ||\sigma_i||_2^2 - \langle \log \sigma_i^2, \mathbf{1} \rangle \right). \quad (7)$$

We manually set the weighting of the KL term relative to the rest of the loss. Proper probabilistic derivation is non-straightforward, and we leave it for future research.

We trained on 227×227 pixel crops of 256×256 pixel ILSVRC-2012 images. The encoder architecture is the same as AlexNet up to layer FC6, and the decoder architecture is shown in Table 5. We initialized the encoder with AlexNet weights, however, this is not necessary, as shown

Figure 4: Samples from VAE with the SE loss (**topmost**) and the proposed DeePSiM loss (**top to bottom:** AlexNet CONV5, AlexNet FC6, VideoNet CONV5).

Type	fc	fc	fc	reshape	uconv	conv
InSize	–	–	–	1	4	8
OutCh	4096	4096	4096	256	256	512
Kernel	–	–	–	–	4	3
Stride	–	–	–	–	↑2	1

Type	uconv	conv	uconv	conv	uconv	uconv	uconv
InSize	8	16	16	32	32	64	128
OutCh	256	256	128	128	64	32	3
Kernel	4	3	4	3	4	4	4
Stride	↑2	1	↑2	1	↑2	↑2	↑2

Table 5: Generator architecture for inverting layer FC6 of AlexNet.

in the appendix. We sampled from the model by sampling the latent variables from a standard Gaussian $z = \varepsilon$ and generating images from that with the decoder.

Samples generated with the usual SE loss, as well as three different comparators (AlexNet CONV5, AlexNet FC6, VideoNet CONV5) are shown in Fig. 4. While Euclidean loss leads to very blurry samples, our method yields images with realistic statistics. Interestingly, the samples trained with the VideoNet comparator look qualitatively similar to the ones with AlexNet, showing that supervised training may not be necessary to yield a good comparator. More results are shown in the appendix.

4.3. Inverting AlexNet

Analysis of learned representations is an important but largely unsolved problem. One approach is to invert the representation. This may give insights into which information is preserved in the representation and what are its invariance properties. However, inverting a non-trivial feature representation Φ, such as the one learned by a large convolutional network, is a difficult ill-posed problem.

Our proposed approach inverts the AlexNet convolutional network very successfully. Surprisingly rich information about the image is preserved in deep layers of the network and even in the predicted class probabilities. While being an interesting result in itself, this also shows how DeePSiM is an excellent loss function when dealing with very difficult image restoration tasks.

Suppose we are given a feature representation Φ, which we aim to invert, and an image $\mathbf{I}$. There are two inverse mappings: Φ_R^{-1} such that $\Phi(\Phi_R^{-1}(\phi)) \approx \phi$, and Φ_L^{-1} such that $\Phi_L^{-1}(\Phi(\mathbf{I})) \approx \mathbf{I}$. Recently two approaches to inversion have been proposed, which correspond to these two variants of the inverse.

Mahendran & Vedaldi (2015), as well as Simonyan et al. (2014) and Yosinski et al. (2015), apply gradient-based

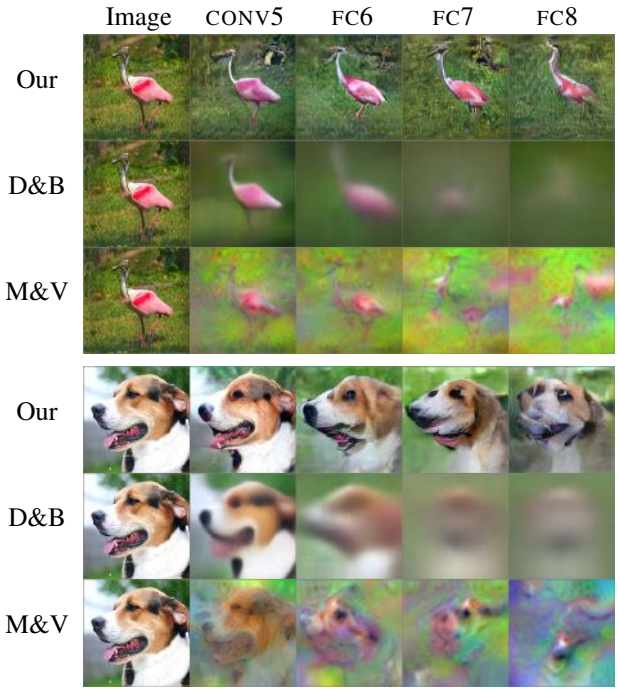

Figure 6: Comparison with Dosovitskiy & Brox (2015) and Mahendran & Vedaldi (2015). Our results look significantly better, even our failure cases (second image).

optimization to find an image $\widetilde{\mathbf{I}}$ which minimizes the loss

$$||\Phi(\mathbf{I}) - \Phi(\widetilde{\mathbf{I}})||_2^2 + P(\widetilde{\mathbf{I}}), \tag{8}$$

where P is a simple natural image prior, such as total variation (TV) regularizer. This method produces images which are roughly natural and have features similar to the input features, corresponding to Φ_R^{-1}. However, the prior is limited, so reconstructions from fully connected layers of AlexNet do not look much like natural images.

Dosovitskiy & Brox (2015) train up-convolutional networks on a large training set of natural images to perform the inversion task. They use SE distance in the image space as loss function, which leads to approximating Φ_L^{-1}. The networks learn to reconstruct the color and rough positions of objects well, but produce over-smoothed results because they average all potential reconstructions.

Our method can be seen as combining the best of both worlds. Loss in the feature space helps preserve perceptually important image features. Adversarial training keeps reconstructions realistic. Note that similar to Dosovitskiy & Brox (2015) and unlike Mahendran & Vedaldi (2015), our method does not require the feature representation being inverted to be differentiable.

Technical details. The generator in this setup takes the features extracted by AlexNet and generates an image from

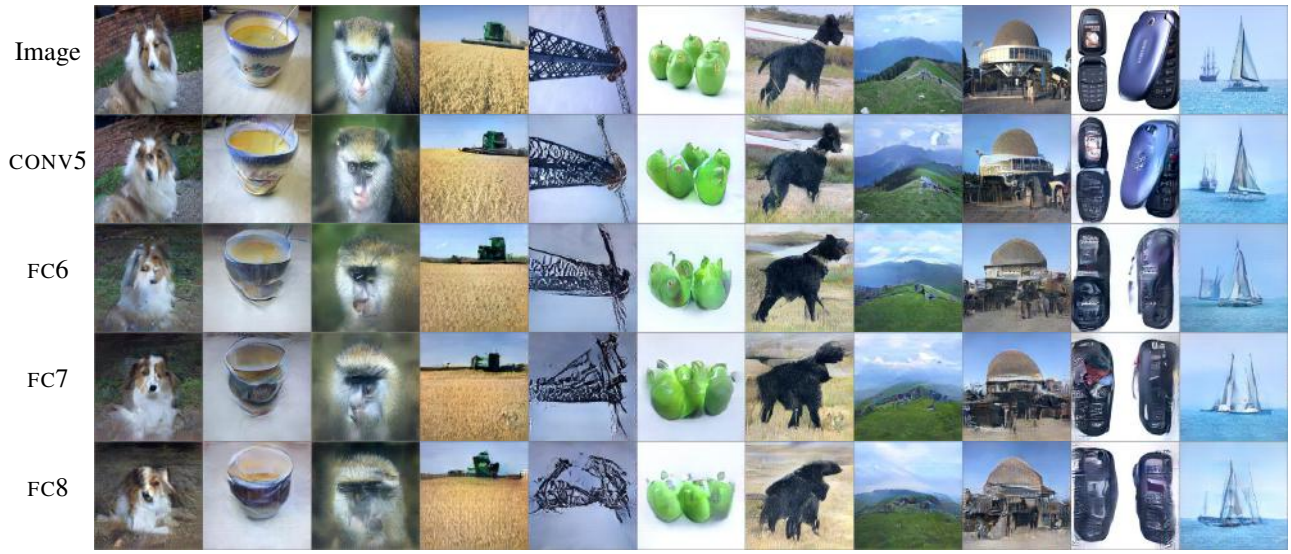

Figure 5: Representative reconstructions from higher layers of AlexNet. General characteristics of images are preserved very well. In some cases (simple objects, landscapes) reconstructions are nearly perfect even from FC8. In the leftmost column the network generates dog images from FC7 and FC8.

them, that is, $\mathbf{x} = \Phi(\mathbf{I}), \mathbf{y} = \mathbf{I}$. In general we followed Dosovitskiy & Brox (2015) in designing the generators. The only modification is that we inserted more convolutional layers, giving the network more capacity. We reconstruct from outputs of layers CONV5 –FC8. In each layer we also include processing steps following the layer, that is, pooling and non-linearities. So for example CONV5 means pooled features (pool5), and FC6 means rectified values (relu6).

Architecture used for inverting FC6 is the same as the decoder of the VAE shown in Table 5. Architectures for other layers are similar, except that for reconstruction from CONV5 fully connected layers are replaced by convolutional ones. The discriminator is the same as used for VAE. We trained on the ILSVRC-2012 training set and evaluated on the ILSVRC-2012 validation set.

Ablation study. We tested if all components of our loss are necessary. Results with some of these components removed are shown in Fig. 7. Clearly the full model performs best. In the following we will give some intuition why.

Training just with loss in the image space leads to averaging all potential reconstructions, resulting in oversmoothed images. One might imagine that adversarial training would allow to make images sharp. This indeed happens, but the resulting reconstructions do not correspond to actual objects originally contained in the image. The reason is that any "natural-looking" image which roughly fits the blurry prediction minimizes this loss. Without the adversarial loss predictions look very noisy. Without the image space loss the method works well, but one can notice artifact on the borders of images, and training was less stable in this case.

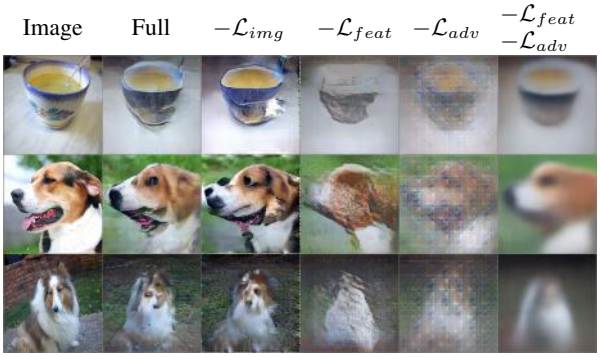

Figure 7: Reconstructions from FC6 with some components of the loss removed.

Sampling pre-images. Given a feature vector ϕ, it would be interesting to sample multiple images $\widetilde{\mathbf{I}}$ such that $\Phi(\widetilde{\mathbf{I}}) = \phi$. A straightforward approach would inject noise into the generator along with the features, so that the network could randomize its outputs. This does not yield the desired result, since nothing in the loss function forces the generator to output multiple different reconstructions per feature vector. A major problem is that in the training data we only have one image per feature vector, i.e., a single sample per conditioning vector. We did not attack this problem in our paper, but we believe it is an important research direction.

	CONV5	FC6	FC7	FC8
Mahendran&Vedaldi	71/19	80/19	82/16	84/09
Dosovitskiy & Brox	35/–	51/–	56/–	58/–
Our just image loss	–/–	46/79	–/–	–/–
Our AlexNet CONV5	43/37	55/48	61/45	63/29
Our VideoNet CONV5	–/–	51/57	–/–	–/–

Figure 8: Normalized inversion error (in %) when reconstructing from different layers of AlexNet with different methods. First in each pair – error in the image space, second – in the feature space.

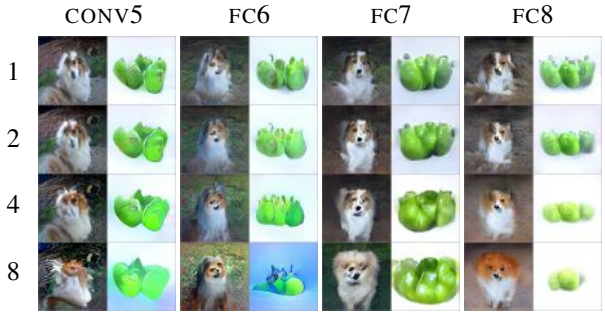

Figure 9: Iteratively re-encoding images with AlexNet and reconstructing. Iteration number shown on the left.

Best results. Representative reconstructions from higher layers of AlexNet are shown in Fig. 5. Comparison with existing approaches is shown in Fig. 6. Reconstructions from CONV5 are near-perfect, combining the natural colors and sharpness of details. Reconstructions from fully connected layers are still very good, preserving the main features of images, colors, and positions of large objects.

Normalized Euclidean error in image space and in feature space (that is, the distance between the features of the image and the reconstruction) are shown in Table 8. The method of Mahendran&Vedaldi performs well in feature space, but not in image space, the method of Dosovitskiy&Brox — vice versa. The presented approach is fairly good on both metrics.

Iterative re-encoding. We performed another experiment illustrating how similar are the features of reconstructions to the original image features. Given an image, we compute its features, generate an image from those, and then iteratively compute the features of the result and generate from those. Results are shown in Fig. 9. Interestingly, several iterations do not significantly change the reconstruction, indicating that important perceptual features are preserved in the generated images. More results are shown in the appendix.

Interpolation. We can morph images into each other by linearly interpolating between their features and generating the corresponding images. Fig. 11 shows that objects shown in the images smoothly warp into each other. More examples are shown in the appendix.

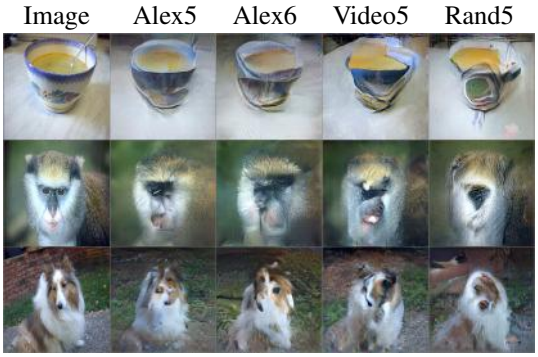

Figure 10: Reconstructions from FC6 with different comparators. The number indicates the layer from which features were taken.

Different comparators. AlexNet network we used above as comparator has been trained on a huge labeled dataset. Is this supervision really necessary to learn a good comparator? We show here results with several alternatives to CONV5 features of AlexNet: 1) FC6 features of AlexNet, 2) CONV5 of AlexNet with random weights, 3) CONV5 of the network of Wang & Gupta (2015) which we refer to as VideoNet.

The results are shown in Fig. 10. While AlexNet CONV5 comparator provides best reconstructions, other networks preserve key image features as well. We also ran preliminary experiments with CONV5 features from the discriminator serving as a comparator, but were not able to get satisfactory results with those.

5. Conclusion

We proposed a class of loss functions applicable to image generation that are based on distances in feature spaces. Applying these to three tasks — image auto-encoding, random natural image generation with a VAE and feature inversion — reveals that our loss is clearly superior to the typical loss in image space. In particular, it allows reconstruction of perceptually important details even from very low-dimensional image representations. We evaluated several feature spaces to measure distances. More research is necessary to find optimal features to be used depending on the task. To control the degree of realism in generated images, an alternative to adversarial training is an approach making use of feature statistics, similar to Gatys et al. (2015). We see these as interesting directions of future work.

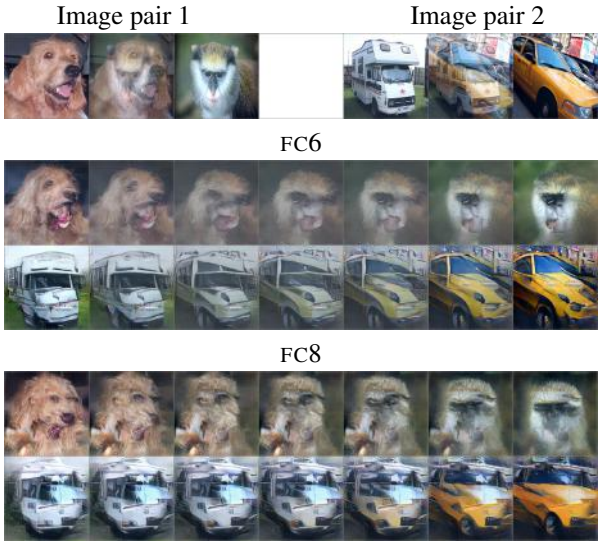

Figure 11: Interpolation between images by interpolating between their features in FC6 and FC8.

Acknowledgements

The authors are grateful to Jost Tobias Springenberg and Philipp Fischer for useful discussions. We acknowledge funding by the ERC Starting Grant VideoLearn (279401).

References

Y. Bengio, E. Laufer, G. Alain, and J. Yosinski. Deep generative stochastic networks trainable by backprop. In *ICML*, 2014.

A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. *AISTATS*, 2011.

S. Daly. Digital images and human vision. chapter The Visible Differences Predictor: An Algorithm for the Assessment of Image Fidelity, pp. 179–206. MIT Press, 1993.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

E. L. Denton, S. Chintala, arthur Szlam, and R. Fergus. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *Advances in Neural Information Processing Systems 28*, pp. 1486–1494. Curran Associates, Inc., 2015.

A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015a.

A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. *arxiv/1506.02753v2*, 2015. URL http://arxiv.org/abs/1506.02753v2.

A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015b.

L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arxiv:1508.06576*, 2015. URL http://arxiv.org/abs/1508.06576.

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.

K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1462–1471, 2015.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313 (5786):504–507, July 2006.

G. E. Hinton and T. J. Sejnowski. Learning and relearning in boltzmann machines. In *Parallel Distributed Processing: Volume 1: Foundations*, pp. 282–317. MIT Press, Cambridge, 1986.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7): 1527–1554, 2006.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

D. Kingma, D. Rezende, S. Mohamed, and M. Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pp. 1106–1114, 2012.

A. B. L. Larsen, S. K. Sønderby, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *arxiv:1512.09300*, 2015. URL http://arxiv.org/abs/1512.09300.

H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, pp. 609–616, 2009.

A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.

M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv:1511.05440*, 2015. URL http://arxiv.org/abs/1511.05440.

M. Mirza and S. Osindero. Conditional generative adversarial nets. *arxiv:1411.1784*, 2014.

A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434*, 2015. URL http://arxiv.org/abs/1511.06434.

K. Ridgeway, J. Snell, B. Roads, R. S. Zemel, and M. C. Mozer. Learning to generate images with perceptual similarity metrics. *arxiv:1511.06409*, 2015.

R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *AISTATS*, 2009.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR workshop track*, 2014. URL http://arxiv.org/abs/1312.6034.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing: Volume 1: Foundations*, pp. 194–281. MIT Press, Cambridge, 1987.

C. J. van den Branden Lambrecht and O. Verscheure. Perceptual quality measure using a spatio-temporal model of the human visual system. *Electronic Imaging: Science & Technology*, 1996.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pp. 1096–1103, 2008.

X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

S. Winkler. A perceptual distortion metric for digital color images. In *in Proc. SPIE*, pp. 175–184, 1998.

J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, ICML*, 2015.

Appendix

Here we show some additional results obtained with the proposed method.

Figure 12 illustrates how position and color of an object is preserved in deep layers of AlexNet (Krizhevsky et al., 2012).

Figure 13 shows results of generating images from interpolations between the features of natural images.

Figure 14 shows samples from variational autoencoders with different losses. Fully unsupervised VAE with VideoNet (Wang & Gupta, 2015) loss and random initialization of the encoder is in the bottom right. Samples from this model are qualitatively similar to others, showing that initialization with AlexNet is not necessary.

Figures 15 and 16 show results of iteratively encoding images to a feature representation and reconstructing back to the image space. As can be seen from Figure 16, the network trained with loss in the image space does not preserve the features well, resulting in reconstructions quickly diverging from the original image.

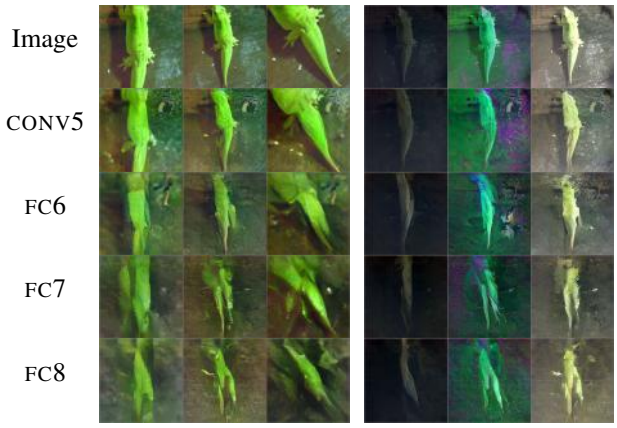

Figure 12: Position (first three columns) and color (last three columns) preservation.

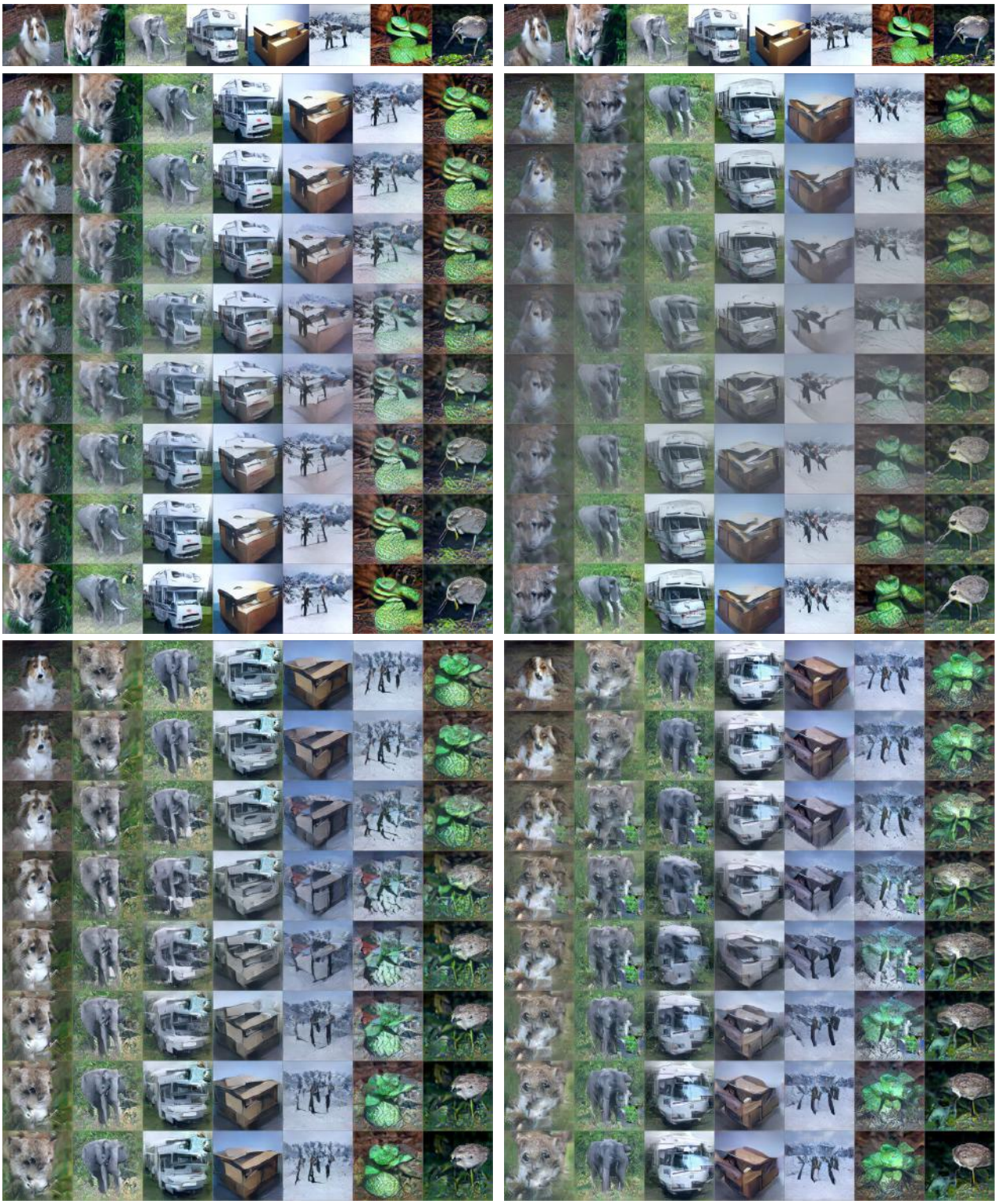

Figure 13: Interpolation in feature spaces at different layers of AlexNet. **Topmost:** input images, **Top left:** CONV5, **Top right:** FC6, **Bottom left:** FC7, **Bottom right:** FC8.

Figure 14: Samples from VAE with our approach, with different comparators. **Top left:** AlexNet CONV5 comparator, **Top right:** AlexNet FC6 comparator, **Bottom left:** VideoNet CONV5 comparator, **Bottom right:** VideoNet CONV5 comparator with randomly initialized encoder.

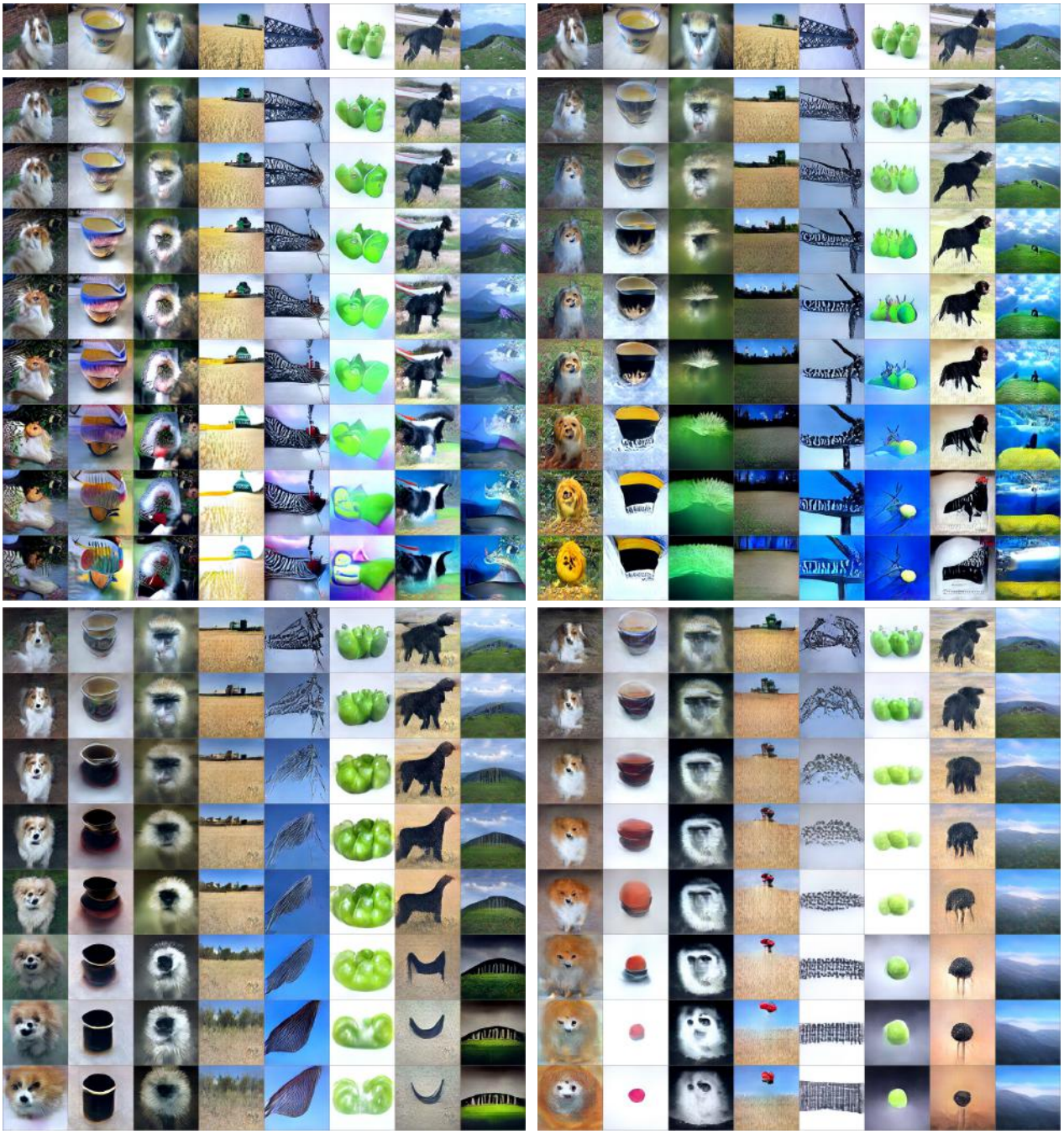

Figure 15: Iterative re-encoding and reconstructions for different layers of AlexNet. Each row of each block corresponds to an iteration number: 1, 2, 4, 6, 8, 12, 16, 20. **Topmost:** input images, **Top left:** CONV5, **Top right:** FC6, **Bottom left:** FC7, **Bottom right:** FC8.

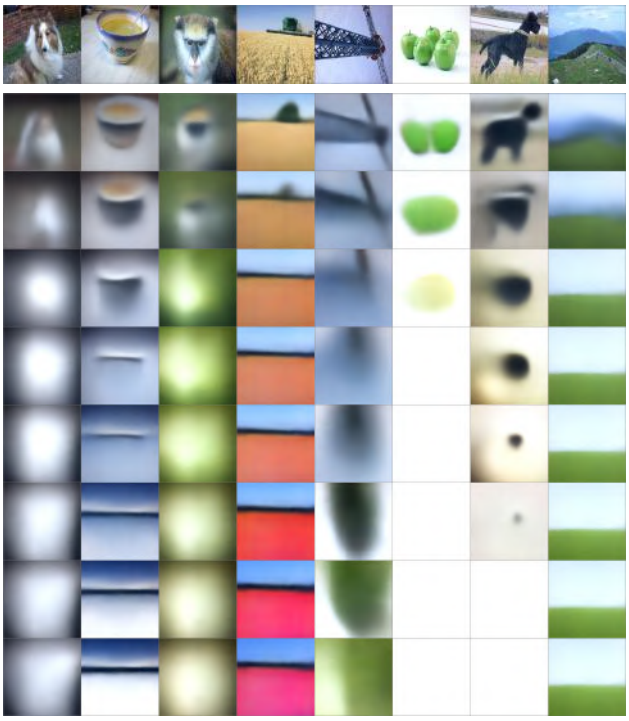

Figure 16: Iterative re-encoding and reconstructions with network trained to reconstruct from AlexNet FC6 layer with squared Euclidean loss in the image space. On top the input images are shown. Then each row corresponds to an iteration number: 1, 2, 4, 6, 8, 12, 16, 20.