The Unreasonable Effectiveness of Deep Features as a Perceptual Metric

Richard Zhang[1] Phillip Isola[12] Alexei A. Efros[1] Eli Shechtman[3] Oliver Wang[3]
[1]UC Berkeley [2]OpenAI [3]Adobe Research
{rich.zhang, isola, efros}@eecs.berkeley.edu {elishe,owang}@adobe.com

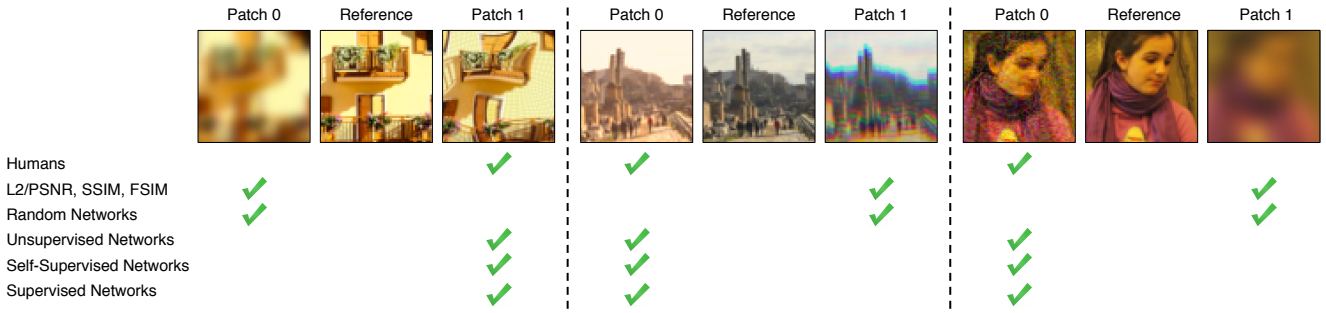


Figure 1: **Which patch (left or right) is "closer" to the middle patch in these examples?** In each case, the traditional metrics (L2/PSNR, SSIM, FSIM) disagree with human judgments. But deep networks, even across architectures (Squeezenet [20], AlexNet [27], VGG [52]) and supervision type (supervised [47], self-supervised [13, 40, 43, 64], and even unsupervised [26]), provide an *emergent embedding* which agrees surprisingly well with humans. We further calibrate existing deep embeddings on a large-scale database of perceptual judgments; models and data can be found at https://www.github.com/richzhang/PerceptualSimilarity.

Abstract

While it is nearly effortless for humans to quickly assess the perceptual similarity between two images, the underlying processes are thought to be quite complex. Despite this, the most widely used perceptual metrics today, such as PSNR and SSIM, are simple, shallow functions, and fail to account for many nuances of human perception. Recently, the deep learning community has found that features of the VGG network trained on ImageNet classification has been remarkably useful as a training loss for image synthesis. But how perceptual are these so-called "perceptual losses"? What elements are critical for their success? To answer these questions, we introduce a new dataset of human perceptual similarity judgments. We systematically evaluate deep features across different architectures and tasks and compare them with classic metrics. We find that deep features outperform all previous metrics by large margins on our dataset. More surprisingly, this result is not restricted to ImageNet-trained VGG features, but holds across different deep architectures and levels of supervision (supervised, self-supervised, or even unsupervised). Our results suggest that perceptual similarity is an emergent property shared across deep visual representations.

1. Motivation

The ability to compare data items is perhaps the most fundamental operation underlying all of computing. In many areas of computer science it does not pose much difficulty: one can use Hamming distance to compare binary patterns, edit distance to compare text files, Euclidean distance to compare vectors, etc. The unique challenge of computer vision is that even this seemingly simple task of comparing visual patterns remains a wide-open problem. Not only are visual patterns very high-dimensional and highly correlated, but, the very notion of visual similarity is often subjective, aiming to mimic human visual perception. For instance, in image compression, the goal is for the compressed image to be indistinguishable from the original by a human observer, irrespective of the fact that their pixel representations might be very different.

Classic per-pixel measures, such as ℓ_2 Euclidean distance, commonly used for regression problems, or the related Peak Signal-to-Noise Ratio (PSNR), are insufficient for assessing structured outputs such as images, as they assume pixel-wise independence. A well-known example is that blurring causes large perceptual but small ℓ_2 change.

What we would really like is a "perceptual distance," which measures how similar are two images in a way that coincides with human judgment. This problem has been a longstanding goal, and there have been numerous perceptually motivated distance metrics proposed, such as SSIM [58], MSSIM [60], FSIM [62], and HDR-VDP [34].

However, constructing a perceptual metric is challenging, because human judgments of similarity (1) depend on high-order image structure [58], (2) are context-dependent

(a) **Traditional** (b) **CNN-based**

Figure 2: **Example distortions.** We show example distortions using our (a) traditional and (b) CNN-based methods.

[19, 36, 35], and (3) may not actually constitute a distance metric [56]. The crux of (2) is that there are many different "senses of similarity" that we can simultaneously hold in mind: is a red circle more similar to a red square or to a blue circle? Directly fitting a function to human judgments may be intractable due the the context-dependent and pairwise nature of the judgments (which compare the similarity between *two* images). Indeed, we show in this paper a negative result where this approach fails to generalize, even when trained on a large-scale dataset containing many distortion types.

Instead, might there be a way to learn a notion of perceptual similarity without directly training for it? The computer vision community has discovered that internal activations of deep convolutional networks, though trained on a high-level image classification task, are often surprisingly useful as a representational space for a much wider variety of tasks. For example, features from the VGG architecture [52] have been used on tasks such as neural style transfer [17], image superresolution [23], and conditional image synthesis [14, 8]. These methods measure distance in VGG feature space as a "perceptual loss" for image regression problems [23, 14].

But how well do these "perceptual losses" actually correspond to human visual perception? How do they compare to traditional perceptual image evaluation metrics? Does the network architecture matter? Does it have to be trained on the ImageNet classification task, or would other tasks work just as well? Do the networks need to be trained at all?

In this paper, we evaluate these questions on a new large-scale database of human judgments, and arrive at several surprising conclusions. We find that internal activations of networks trained for high-level classification tasks, even across network architectures [20, 28, 52] and no further calibration, do indeed correspond to human perceptual judgments. In fact, they correspond far better than the commonly used metrics like SSIM and FSIM [58, 62], which were not designed to handle situations where spatial ambiguities are a factor [49]. Furthermore, the best performing self-supervised networks, including BiGANs [13], cross-channel prediction [64], and puzzle solving [40] perform just as well at this task, even without the benefit of human-labeled training data. Even a simple unsupervised network initialization with stacked k-means [26] beats the classic metrics by a large margin! This illustrates an *emergent property* shared across networks, even across architectures and training signals. Importantly, however, having *some* training signal appears crucial – a randomly initialized network achieves much lower performance.

Our study is based on a newly collected perceptual similarity dataset, using a large set of distortions and real algorithm outputs. It contains both traditional distortions, such as contrast and saturation adjustments, noise patterns, filtering, and spatial warping operations, and CNN-based algorithm outputs, such as autoencoding, denoising, and colorization, produced by a variety of architectures and losses. Our dataset is richer and more varied than previous datasets of this kind [45]. We also collect judgments on outputs from real algorithms for the tasks of superresolution, frame interpolation, and image deblurring, which is especially important as these are the real-world use cases for a perceptual metric. We show that our data can be used to "calibrate" existing networks, by learning a simple linear scaling of layer activations, to better match low-level human judgments.

Our results are consistent with the hypothesis that perceptual similarity is not a special function all of its own, but rather a *consequence* of visual representations tuned to be predictive about important structure in the world. Representations that are effective at semantic prediction tasks are also representations in which Euclidean distance is highly predictive of perceptual similarity judgments.

Our contributions are as follows:

- We introduce a large-scale, highly varied, perceptual similarity dataset, containing 484k human judgments. Our dataset not only includes parameterized distortions, but also real algorithm outputs. We also collect judgments on a different perceptual test, just noticeable differences (JND).
- We show that deep features, trained on supervised, self-supervised, and unsupervised objectives alike,

Dataset	# Input Imgs/ Patches	Input Type	Num Distort.	Distort. Types	# Levels	# Distort. Imgs/Patches	# Judgments	Judgment Type
LIVE [51]	29	images	5	traditional	continuous	.8k	25k	MOS
CSIQ [29]	30	images	6	traditional	5	.8k	25k	MOS
TID2008 [46]	25	images	17	traditional	4	2.7k	250k	MOS
TID2013 [45]	25	images	24	traditional	5	3.0k	500k	MOS
BAPPS (2AFC–Distort)	160.8k	64×64 patch	425	trad + CNN	continuous	321.6k	349.8k	2AFC
BAPPS (2AFC–Real alg)	26.9k	64×64 patch	–	alg outputs	–	53.8k	134.5k	2AFC
BAPPS (JND–Distort)	9.6k	64×64 patch	425	trad. + CNN	continuous	9.6k	28.8k	Same/Not same

Table 1: **Dataset comparison.** A primary differentiator between our proposed Berkeley-Adobe Perceptual Patch Similarity (BAPPS) dataset and previous work is scale of distortion types. We provide human perceptual judgments on distortion set using uncompressed images from [7, 10]. Previous datasets have used a small number of distortions at discrete levels. We use a large number of distortions (created by sequentially composing atomic distortions together) and sample continuously. For each input patch, we corrupt it using two distortions and ask for a few human judgments (2 for train, 5 for test set) per pair. This enables us to obtain judgments on a large number of patches. Previous databases summarize their judgments into a mean opinion score (MOS); we simply report pairwise judgments (two alternative force choice). In addition, we provide judgments on outputs from *real algorithms*, as well as a same/not same Just Noticeable Difference (JND) perceptual test.

model low-level perceptual similarity surprisingly well, outperforming previous, widely-used metrics.

- We demonstrate that network architecture alone does not account for the performance: untrained nets achieve much lower performance.
- With our data, we can improve performance by "calibrating" feature responses from a pre-trained network.

Prior work on datasets In order to evaluate existing similarity measures, a number of datasets have been proposed. Some of the most popular are the LIVE [51], TID2008 [46], CSIQ [29], and TID2013 [45] datasets. These datasets are referred to Full-Reference Image Quality Assessment (FR-IQA) datasets and have served as the de-facto baselines for development and evaluation of similarity metrics. A related line of work is on No-Reference Image Quality Assessment (NR-IQA), such as AVA [38] and LIVE In the Wild [18]. These datasets investigate the "quality" of individual images by themselves, without a reference image. We collect a new dataset that is complementary to these: it contains a substantially larger number of distortions, including some from newer, deep network based outputs, as well as geometric distortions. Our dataset is focused on perceptual similarity, rather than quality assessment. Additionally, it is collected on patches as opposed to full images, in the wild, with a different experimental design (more details in Sec 2).

Prior work on deep networks and human judgments Recently, advances in DNNs have motivated investigation of applications in the context of visual similarity and image quality assessment. Kim and Lee [25] use a CNN to predict visual similarity by training on low-level differences. Concurrent work by Talebi and Milanfar [54, 55] train a deep network in the context of NR-IQA for image aesthetics. Gao et al. [16] and Amirshahi et al. [3] propose techniques involving leveraging internal activations of deep networks (VGG and AlexNet, respectively) along with additional multiscale post-processing. In this work, we conduct a more in-depth study across different architectures, training signals, on a new, large scale, highly-varied dataset.

Recently, Berardino et al. [6] train networks on perceptual similarity, and importantly, assess the ability of deep networks to make predictions on a *separate* task – predicting most and least perceptually-noticeable directions of distortion. Similarly, we not only assess image patch similarity on parameterized distortions, but also test generalization to real algorithms, as well as generalization to a separate perceptual task – just noticeable differences.

2. Berkeley-Adobe Perceptual Patch Similarity (BAPPS) Dataset

To evaluate the performance of different perceptual metrics, we collect a large-scale highly diverse dataset of perceptual judgments using two approaches. Our main data collection employs a two alternative forced choice (2AFC) test, that asks which of two distortions is more similar to a reference. This is validated by a second experiment where we perform a just noticeable difference (JND) test, which asks whether two patches – one reference and one distorted – are the same or different. These judgments are collected over a wide space of distortions and real algorithm outputs.

2.1. Distortions

Traditional distortions We create a set of "traditional" distortions consisting of common operations performed on the input patches, listed in Table 2 (left). In general, we use photometric distortions, random noise, blurring, spatial shifts and corruptions, and compression artifacts. We show qualitative examples of our traditional distortions in Figure 2. The severity of each perturbation is parameterized -

Sub-type	Distortion type
Photometric	lightness shift, color shift, contrast, saturation
Noise	uniform white noise, Gaussian white, pink, & blue noise, Gaussian colored (between violet and brown) noise, checkerboard artifact
Blur	Gaussian, bilateral filtering
Spatial	shifting, affine warp, homography, linear warping, cubic warping, ghosting, chromatic aberration,
Compression	jpeg

Parameter type	Parameters
Input corruption	null, pink noise, white noise, color removal, downsampling
Generator network architecture	# layers, # skip connections, # layers with dropout, force skip connection at highest layer, upsampling method, normalization method, first layer stride # channels in 1^{st} layer, max # channels
Discriminator	number of layers
Loss/Learning	weighting on oixel-wise (ℓ_1), VGG, discriminator losses, learning rate

Table 2: **Our distortions.** Our traditional distortions (left) are performed by basic low-level image editing operations. We also sequentially compose them to better explore the space. Our CNN-based distortions (right) are formed by randomly varying parameters such as task, network architecture, and learning parameters. The goal of the distortions is to mimic plausible distortions seen in real algorithm outputs.

for example, for Gaussian blur, the kernel width determines the amount of corruption applied to the input image. We also compose pairs of distortions sequentially to increase the overall space of possible distortions. In total, we have 20 distortions and 308 sequentially composed distortions.

CNN-based distortions To more closely simulate the space of artifacts that can arise from deep-learning based methods, we create a set of distortions created by neural networks. We simulate possible algorithm outputs by exploring a variety of tasks, architectures, and losses, as shown in Table 2 (right). Such tasks include autoencoding, denoising, colorization, and superresolution. All of these tasks can be achieved by applying the appropriate corruption to the input. In total, we generated 96 "denoising autoencoders" and use these as CNN-based distortion functions. We train each of these networks on the 1.3M ImageNet dataset [47] for 1 epoch. The goal of each network is not to solve the task per se, but rather to explore common artifacts that plague the outputs of deep learning based methods.

Distorted image patches from real algorithms The true test of an image assessment algorithm is on real problems and real algorithms. We gather perceptual judgments using such outputs. Data on real algorithms is more limited, as each application will have their own unique properties. For example, different colorization methods will not show much structural variation, but will be prone to effects such as color bleeding and color variation. On the other hand, superresolution will not have color ambiguity, but may see larger structural changes from algorithm to algorithm.

Superresolution We evaluate results from the 2017 NTIRE workshop [2]. We use 3 tracks from the workshop – $\times 2$, $\times 3$, $\times 4$ upsampling rates using "unknown" downsampling to create the input images. Each track had approximately 20 algorithm submissions. We also evaluate several additional methods, including bicubic upsampling, and four of the top performing deep superresolution methods [24, 59, 31, 48]. A common qualitative way of presenting superresolution results is zooming into specific patches and comparing differences. As such, we sample random 64×64 triplets from random locations of images in the Div2K [2] dataset – the ground truth high-resolution image, along with two algorithm outputs.

Frame interpolation We sample patches from different frame interpolation algorithms, including three variants of flow-based interpolation [33], CNN-based interpolation [39], and phase-based interpolation [37] on the Davis Middleburry dataset [50]. Because artifacts arising from frame interpolation may occur at different scales, we randomly rescale the image before sampling a patch triplet.

Video deblurring We sample from the video deblurring dataset [53], along with deblurring outputs from Photoshop Shake Reduction, Weighted Fourier Aggregation [11], and three variants of a deep video deblurring method [53].

Colorization We sample patches using random scales on the colorization task, on images from the ImageNet dataset [47]. The algorithms are from pix2pix [22], Larsson et al. [30], and variants from Zhang et al. [63].

2.2. Psychophysical Similarity Measurements

2AFC similarity judgments We randomly select an image patch x and apply two distortions to produce patches x_0, x_1. We then ask a human which is closer to the original patch x, and record response $h \in \{0, 1\}$. On average, people spent approximately 3 seconds per judgment. Let $\mathcal{T}$ denote our dataset of patch triplets (x, x_0, x_1, h).

A comparison between our dataset and previous datasets is shown in Table 1. Previous datasets have focused on collecting large numbers of human judgments for a few images and distortion types. For example, the largest dataset, TID2013 [45], has 500k judgments on 3000 distortions (from 25 input images with 24 distortions types, each sampled at 5 levels). We provide a complementary dataset that focuses instead on a large number of distortions types. In, addition, we collect judgments on a large number of

64×64 patches rather than a small number of images. There are three reasons for this. First, the space of full images is extremely large, which makes it much harder to cover a reasonable portion of the domain with judgments (even 64×64 color patches represent an intractable 12k-dimensional space). Second, by choosing a smaller patch size, we focus on lower-level aspects of similarity, to mitigate the effect of differing "respects of similarity" that may be influenced by high-level semantics [36]. Finally, modern methods for image synthesis train deep networks with patch-based losses (implemented as convolutions) [8, 21]. Our dataset consists of over 161k patches, derived from the MIT-Adobe 5k dataset [7] (5000 uncompressed images) for training, and the RAISE1k dataset [10] for validation.

To enable large-scale collection, our data is collected "in-the-wild" on Amazon Mechanical Turk, as opposed to a controlled lab setting. Crump et al. [9] show that AMT can be reliably used to replicate many psychophysics studies, despite the inability to control all environmental factors. We ask for 2 judgments per example in our "train" set and 5 judgments in our "val" sets. Asking for fewer judgments enables us to explore a larger set of image patches and distortions. We add sentinels which consist of pairs of patches with obvious deformations, e.g., a large amount of Gaussian noise vs a small amount of Gaussian noise. Approximately 90% of Turkers were able to correctly pass at least 93% of the sentinels (14 of 15), indicating that they understood the task and were paying attention. We choose to use a larger number of distortions than prior datasets.

Just noticeable differences (JND)

A potential shortcoming of the 2AFC task is that it is "cognitively penetrable," in the sense that participants can consciously choose which respects of similarity they will choose to focus on in completing the task [36], which introduces subjectivity into the judgments. To validate that the judgments actually reflected something objective and meaningful, we also collected user judgments of "just noticeable differences" (JNDs). We show a reference image, followed by a randomly distorted image, and ask a human if the images are the same or different. The two image patches are shown for 1 second each, with a 250 ms gap in between. Two images which look similar may be easily confused, and a good perceptual metric will be able to order pairs from most to least confusable. JND tests like this may be considered less subjective, since there is a single correct answer for each judgment, and participants are presumed to be aware of what correct behavior entails. We gather 3 JND observations for each of the 4.8k patches in our traditional and CNN-based validation sets. Each subject is shown 160 pairs, along with 40 sentinels (32 identical and 8 with large Gaussian noise distortion applied). We also provide a short training period of 10 pairs which contain 4 "same" pairs, 1 obviously different pair, and 5 "different" pairs generated by our distortions. We chose to do this in order to prime the users towards expecting approximately 40% of the patch pairs to be identical. Indeed, 36.4% of the pairs were marked "same" (70.4% of sentinels and 27.9% of test pairs).

Dataset	Data source	Train/ Val	# Examples	# Judge /Example
Traditional	[7]	Train	56.6k	2
CNN-based	[7]	Train	38.1k	2
Mixed	[7]	Train	56.6k	2
2AFC–Distort [Trn]	–	Train	151.4k	2
Traditional	[10]	Train	4.7k	5
CNN-based	[10]	Train	4.7k	5
2AFC–Distort [Val]	–	Val	9.4k	5
Superres	[32]	Val	10.9k	5
Frame Interp	[50]	Val	1.9	5
Video Deblur	[5]	Val	9.4	5
Colorization	[47]	Val	4.7	5
2AFC–Real Alg [Val]	–	Val	26.9k	5
Traditional	[10]	Val	4.8k	3
CNN-based	[10]	Val	4.8k	3
JND–Distort	–	Val	9.6k	3

Table 3: **Our dataset breakdown**. We split our 2AFC dataset in to three main portions (1,2) training and test sets with our distortions. Our training and test sets contain patches sampled from the MIT5k [7] and RAISE1k [10] datasets, respectively (3) a test set containing real algorithm outputs, containing patches from a variety of applications. Our JND data is on traditional and CNN-based distortions.

3. Deep Feature Spaces

We evaluate feature distances in different networks. For a given convolutional layer, we compute cosine distance (in the channel dimension) and average across spatial dimensions and layers of the network. We also discuss how to tune an existing network on our data.

Network architectures We evaluate the SqueezeNet [20], AlexNet [28], and VGG [52] architectures. We use 5 `conv` layers from the VGG network, which has become the de facto standard for image generation tasks [17, 14, 8]. We also compare against the shallower AlexNet network, which may more closely match the architecture of the human visual cortex [61]. We use the `conv1-conv5` layers from [27]. Finally, the SqueezeNet architecture was designed to be extremely lightweight (2.8 MB) in size, with similar classification performance to AlexNet. We use the first `conv` layer and some subsequent "`fire`" modules.

We additionally evaluate self-supervised methods, including puzzle-solving [40], cross-channel prediction [63, 64], learning from video [43], and generative model-

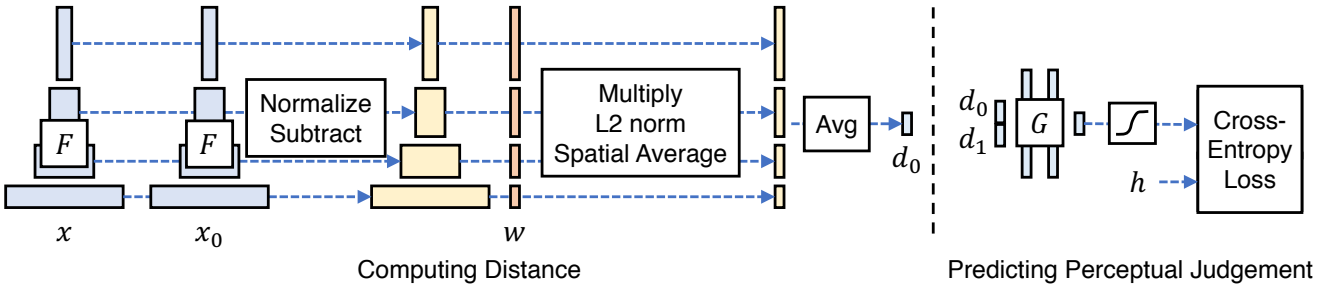


Figure 3: **Computing distance from a network** (Left) To compute a distance d_0 between two patches, x, x_0, given a network $\mathcal{F}$, we first compute deep embeddings, normalize the activations in the channel dimension, scale each channel by vector w, and take the ℓ_2 distance. We then average across spatial dimension and across all layers. (Right) A small network $\mathcal{G}$ is trained to predict perceptual judgment h from distance pair (d_0, d_1).

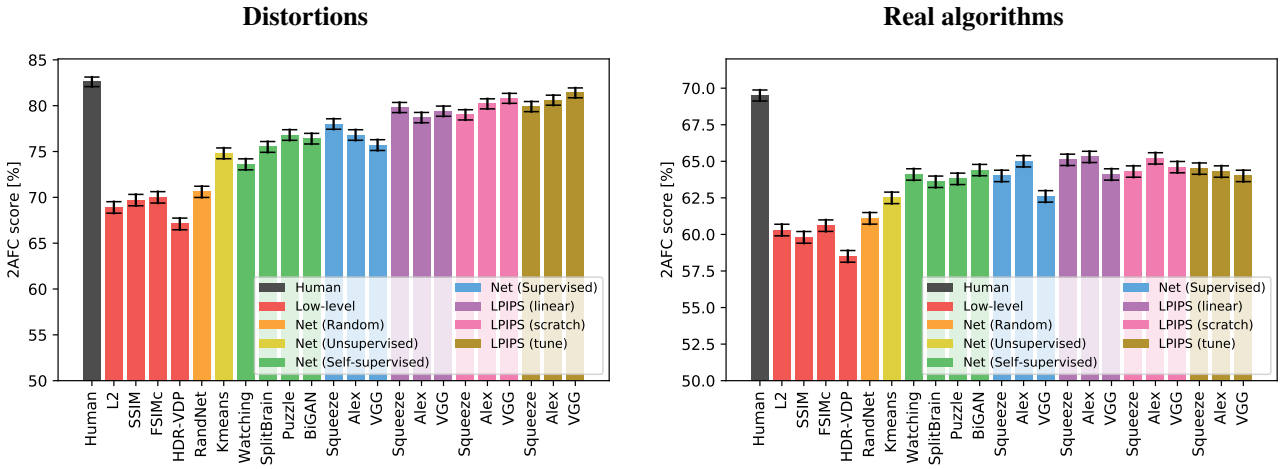


Figure 4: **Quantitative comparison.** We show a quantitative comparison across metrics on our test sets. (Left) Results averaged across our traditional and CNN-based distortions. (Right) Results averaged across our 4 real algorithm sets.

ing [13]. We use publicly available networks from these and other methods, which use variants of AlexNet [28].

Network activations to distance Figure 3 (left) and Equation 1 illustrate how we obtain the distance between reference and distorted patches x, x_0 with network $\mathcal{F}$. We extract feature stack from L layers and unit-normalize in the channel dimension, which we designate as $\hat{y}^l, \hat{y}^l_0 \in \mathbb{R}^{H_l \times W_l \times C_l}$ for layer l. We scale the activations channel-wise by vector $w^l \in \mathbb{R}^{C_l}$ and compute the ℓ_2 distance. Finally, we average spatially and sum channel-wise. Note that using $w_l = 1 \forall l$ is equivalent to computing cosine distance.

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} ||w_l \odot (\hat{y}^l_{hw} - \hat{y}^l_{0hw})||_2^2 \quad (1)$$

Training on our data We consider a few variants for training with our perceptual judgments: ***lin***, ***tune***, and ***scratch***. For the ***lin*** configuration, we keep pre-trained network weights $\mathcal{F}$ fixed, and learn linear weights w on top. This constitutes a "perceptual calibration" of a few parameters in an existing feature space. For example, for the VGG network, 1472 parameters are learned. For the ***tune*** configuration, we initialize from a pre-trained classification model, and allow all the weights for network $\mathcal{F}$ to be fine-tuned. Finally, for ***scratch***, we initialize the network from random Gaussian weights and train it entirely on our judgments. Overall, we refer to these as variants of our proposed **Learned Perceptual Image Patch Similarity (LPIPS)** metric. We illustrate the training loss function in Figure 3 (right) and describe it further in the appendix.

4. Experiments

Results on our validation sets are shown in Figure 4. We first evaluate how well our metrics and networks work. All validation sets contain 5 pairwise judgments for each triplet. Because this is an inherently noisy process, we compute agreement of an algorithm with *all* of the judgments. For example, if there are 4 preferences for x_0 and 1 for x_1, an algorithm which predicts the more popular choice x_0 would receive 80% credit. If a given example is scored with fraction p humans in one direction and $1 - p$ in the other, a human would achieve score $p^2 + (1 - p)^2$ on expectation.

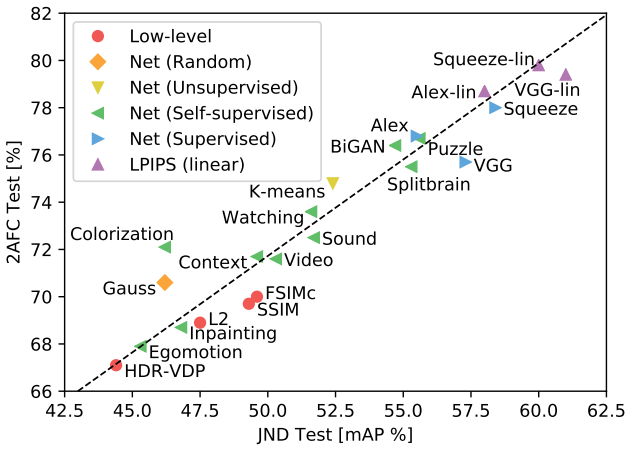


Figure 5: **Correlating Perceptual Tests.** We show performance across methods, including unsupervised [26], self-supervised [1, 44, 12, 57, 63, 41, 43, 40, 13, 64], supervised [27, 52, 20], and our perceptually-learned metrics (LPIPS). The scores are on our 2AFC and JND tests, averaged across traditional and CNN-based distortions.

4.1. Evaluations

How well do low-level metrics and classification networks perform? Figure 4 shows the performance of various low-level metrics (in red), deep networks, and human ceiling (in black). The scores are averaged across the 2 distortion test sets (traditional+CNN-based) in Figure 4 (left), and 4 real algorithm benchmarks (superresolution, frame interpolation, video deblurring, colorization) in Figure 4 (right). All scores within each test set are shown in the appendix. Averaged across all 6 test sets, humans are 73.9% consistent. Interestingly, the supervised networks perform at about the same level to each other, at 68.6%, 68.9%, and 67.0%, even across variation in model sizes – SqueezeNet (2.8 MB), AlexNet (9.1 MB), and VGG (58.9 MB) (only convolutional layers are counted). They all perform better than traditional metrics ℓ_2, SSIM, and FSIM at 63.2%, 63.1%, 63.8%, respectively. Despite its common use, SSIM was not designed for situations where geometric distortion is a large factor [49].

Does the network have to be trained on classification? In Figure 4, we show model performance across a variety of unsupervised and self-supervised tasks, shown in green – generative modeling with BiGANs [13], solving puzzles [40], cross-channel prediction [64], and segmenting foreground objects from video [43]. These self-supervised tasks perform on par with classification networks. This indicates that tasks across a large spectrum can induce representations which transfer well to perceptual distances. Also, the performance of the stacked k-means method [26], shown in yellow, outperforms low-level metrics. Random networks, shown in orange, with weights drawn from a Gaussian, do not yield much improvement. This indicates that the combination of network structure, along with orienting filters in directions where data is more dense, can better correlate to perceptual judgments.

		2AFC	JND	Class.	Det.	Avg
Perceptual	**2AFC**	–	.928	.640	.363	.644
Perceptual	**JND**	.928	–	.612	.232	.591
PASCAL	**Classification**	.640	.612	–	.429	.560
PASCAL	**Detection**	.363	.232	.429	–	.341

Table 4: **Task correlation**. We correlate scores between our low-level perceptual tests along with high-level semantic tests across methods. Perceptual scores are averaged between traditional and CNN-based distortion sets. Correlation scores are computed for AlexNet-like architectures.

In Table 5, we explore how well our perceptual task correlates to semantic tasks on the PASCAL dataset [15], using results summarized in [64], including additional self-supervised methods [1, 44, 12, 57, 63, 41]. We compute the correlation coefficient between each task (perceptual or semantic) across different methods. The correlation from our 2AFC distortion preference task to classification and detection is .640 and .363, respectively. Interestingly, this is similar to the correlation between the classification and detection tasks (.429), even though both are considered "high-level" semantic tasks, and our perceptual task is "low-level."

Do metrics correlate across different perceptual tasks? We test if training for the 2AFC distortion preference test corresponds with another perceptual task, the JND test. We order patch pairs by ascending order by a given metric, and compute precision-recall on our CNN-based distortions – for a good metric, patches which are close together are more likely to be confused for being the same. We compute area under the curve, known as mAP [15]. The 2AFC distortion preference test has high correlation to JND: $\rho = .928$ when averaging the results across distortion types. Figure 5 shows how different methods perform under each perceptual test. This indicates that 2AFC generalizes to another perceptual test and is giving us signal regarding human judgments.

Can we train a metric on traditional and CNN-based distortions? In Figure 4, we show performance using our ***lin***, ***scratch***, and ***tune*** configurations, shown in purple, pink, and brown, respectively. When validating on the traditional and CNN-based distortions (Figure 4(a)), we see improvements. Allowing the network to tune all the way through (brown) achieves higher performance than simply learning linear weights (purple) or training from scratch (pink). The higher capacity network VGG also performs better than the lower capacity SqueezeNet and AlexNet architectures. These results verify that networks can indeed learn from perceptual judgments.

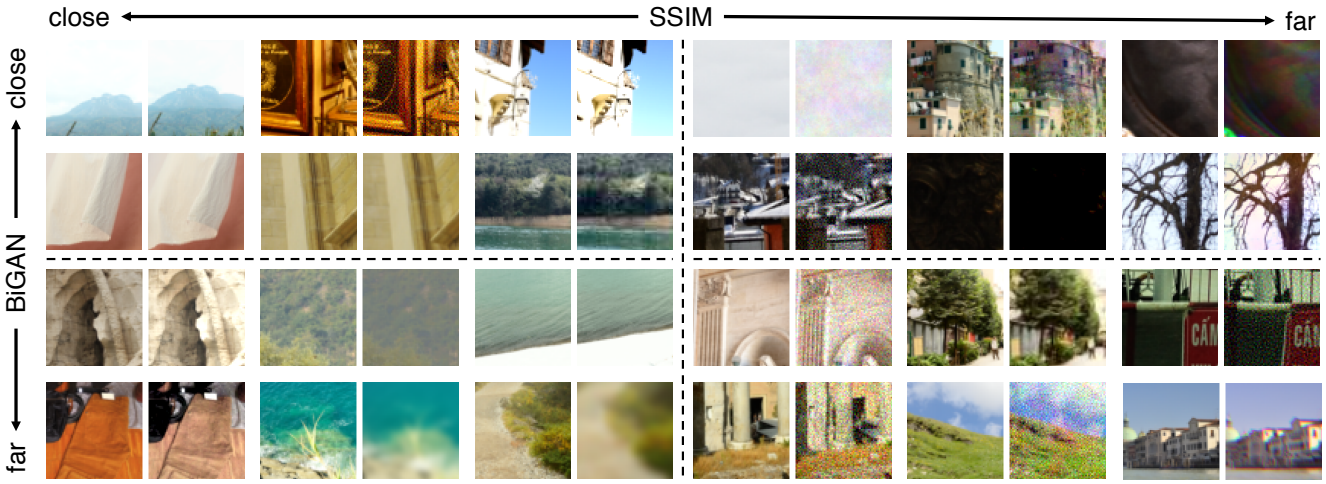


Figure 6: **Qualitative comparisons on distortions.** We show qualitative comparison on traditional distortions, using the SSIM [58] metric and BiGAN network [13]. We show examples where the metrics agree and disagree. A primary difference is that deep embeddings appear to be more sensitive to blur. Please see the appendix for additional examples.

Does training on traditional and CNN-based distortions transfer to real-world scenarios? We are more interested in how performance generalizes to *real-world algorithms*, shown in Figure 4(b). The SqueezeNet, AlexNet, and VGG architectures start at 64.0%, 65.0%, and 62.6%, respectively. Learning a linear classifier (purple) improves performance for all networks. Across the 3 networks and 4 real-algorithm tasks, 11 of the 12 scores improved, indicating that "calibrating" activations on a pre-existing representation using our data is a safe way to achieve a small boost in performance (1.1%, 0.3%, and 1.5%, respectively). Training a network from scratch (pink) yields slightly lower performance for AlexNet, and slightly higher performance for VGG than linear calibration. However, these still outperform low-level metrics. This indicates that the distortions we have expressed do project onto our test-time tasks of judging real algorithms.

Interestingly, starting with a pre-trained network and tuning throughout *lowers* transfer performance. This is an interesting negative result, as training for a low-level perceptual task directly does not necessarily perform as well as transferring a representation trained for the high-level task.

Where do deep metrics and low-level metrics disagree? In Figure 11, we show a qualitative comparison across our traditional distortions for a deep method, BiGANs [13], and a representation traditional perceptual method, SSIM [58]. Pairs which BiGAN perceives to be far but SSIM to be close generally contain some blur. BiGAN tends to perceive correlated noise patterns to be a smaller distortion than SSIM.

5. Conclusions

Our results indicate that networks trained to solve challenging visual prediction and modeling tasks end up learning a representation of the world that correlates well with perceptual judgments. A similar story has recently emerged in the representation learning literature: networks trained on self-supervised and unsupervised objectives end up learning a representation that is also effective at semantic tasks [12]. Interestingly, recent findings in neuroscience make much the same point: representations trained on computer vision tasks also end up being effective models of neural activity in macaque visual cortex [61]. Moreover (and roughly speaking), the stronger the representation is at the computer vision task, the stronger it is as a model of cortical activity. Our paper makes a similar finding: the stronger a feature set is at classification and detection, the stronger it is as a model of perceptual similarity judgments, as suggested in Table 4. Together, these results suggest that a good feature is a good feature. Features that are good at semantic tasks are also good at self-supervised and unsupervised tasks, and also provide good models of both human perceptual behavior and macaque neural activity. This last point aligns with the "rational analysis" explanation of visual cognition [4], suggesting that the idiosyncrasies of biological perception arise as a consequence of a rational agent attempting to solve natural tasks. Further refining the degree to which this is true is an important question for future research.

Acknowledgements This research was supported, in part, by grants from Berkeley Deep Drive, NSF IIS-1633310, and hardware donations by NVIDIA. We thank members of the Berkeley AI Research Lab and Adobe Research for helpful discussions. We thank Alan Bovik for his insightful comments. We also thank Radu Timofte, Zhaowen Wang, Michael Waechter, Simon Niklaus, and Sergio Guadarrama for help preparing data. RZ is partially supported by an Adobe Research Fellowship and much of this work was done while RZ was an intern at Adobe Research.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, pages 37–45, 2015. 7

[2] E. Agustsson and R. Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshops*, July 2017. 4

[3] S. Ali Amirshahi, M. Pedersen, and S. X. Yu. Image quality assessment by comparing cnn features between images. *Electronic Imaging*, 2017(12):42–51, 2017. 3

[4] J. R. Anderson. *The adaptive character of thought*. Psychology Press, 1990. 8

[5] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2011. 5

[6] A. Berardino, V. Laparra, J. Ballé, and E. Simoncelli. Eigen-distortions of hierarchical representations. In *NIPS*, 2017. 3

[7] V. Bychkovsky, S. Paris, E. Chan, and F. Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *CVPR*, 2011. 3, 5

[8] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. *ICCV*, 2017. 2, 5

[9] M. J. Crump, J. V. McDonnell, and T. M. Gureckis. Evaluating amazon's mechanical turk as a tool for experimental behavioral research. *PloS one*, 2013. 5

[10] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato. Raise: a raw images dataset for digital image forensics. In *Proceedings of the 6th ACM Multimedia Systems Conference*, pages 219–224. ACM, 2015. 3, 5

[11] M. Delbracio and G. Sapiro. Hand-held video deblurring via efficient fourier aggregation. *IEEE Transactions on Computational Imaging*, 1(4):270–283, 2015. 4

[12] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 7, 8

[13] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *ICLR*, 2017. 1, 2, 5, 7, 8, 12, 14

[14] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *HIPS*, pages 658–666, 2016. 2, 5

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 7

[16] F. Gao, Y. Wang, P. Li, M. Tan, J. Yu, and Y. Zhu. Deepsim: Deep similarity for image quality assessment. *Neurocomputing*, 2017. 3

[17] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, pages 2414–2423, 2016. 2, 5

[18] D. Ghadiyaram and A. C. Bovik. Massive online crowdsourced study of subjective and objective picture quality. *TIP*, 2016. 3

[19] N. Goodman. Seven strictures on similarity. *Problems and Projects*, 1972. 2

[20] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size. *CVPR*, 2017. 1, 2, 5, 7, 12

[21] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 5

[22] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Learning visual groups from co-occurrences in space and time. *ICCV Workshop*, 2016. 4

[23] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *ECCV*, 2016. 2

[24] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654, 2016. 4

[25] J. Kim and S. Lee. Deep learning of human visual sensitivity in image quality assessment framework. In *CVPR*, 2017. 3

[26] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *International Conference on Learning Representations*, 2016. 1, 2, 7, 12

[27] A. Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014. 1, 5, 7, 12, 14

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2, 5, 6

[29] E. C. Larson and D. M. Chandler. Most apparent distortion: full-reference image quality assessment and the role of strategy. *Journal of Electronic Imaging*, 2010. 3

[30] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. *ECCV*, 2016. 4

[31] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *CVPR*, 2016. 4

[32] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017. 5

[33] C. Liu et al. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, 2009. 4

[34] R. Mantiuk, K. J. Kim, A. G. Rempel, and W. Heidrich. Hdr-vdp-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions. In *ACM Transactions on Graphics (TOG)*, 2011. 1, 12

[35] A. B. Markman and D. Gentner. Nonintentional similarity processing. *The new unconscious*, pages 107–137, 2005. 2

[36] D. L. Medin, R. L. Goldstone, and D. Gentner. Respects for similarity. *Psychological review*, 100(2):254, 1993. 2, 5

[37] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung. Phase-based frame interpolation for video. In *CVPR*, pages 1410–1418, 2015. 4

[38] N. Murray, L. Marchesotti, and F. Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *CVPR*, 2012. 3

[39] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive separable convolution. In *ICCV*, 2017. 4

[40] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *ECCV*, 2016. 1, 2, 5, 7, 12

[41] A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman. Visually indicated sounds. *CVPR*, 2016. 7

[42] A. Paszke, S. Chintala, R. Collobert, K. Kavukcuoglu, C. Farabet, S. Bengio, I. Melvin, J. Weston, and J. Marithoz. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration, may 2017. 13

[43] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. *CVPR*, 2017. 1, 5, 7, 12

[44] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 7

[45] N. Ponomarenko, L. Jin, O. Ieremeiev, V. Lukin, K. Egiazarian, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, et al. Image database tid2013: Peculiarities, results and perspectives. *Signal Processing: Image Communication*, 2015. 2, 3, 4, 10, 14

[46] N. Ponomarenko, V. Lukin, A. Zelensky, K. Egiazarian, M. Carli, and F. Battisti. Tid2008-a database for evaluation of full-reference visual quality assessment metrics. *Advances of Modern Radioelectronics*, 2009. 3

[47] 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. *IJCV*, 2015. 1, 4, 5

[48] M. S. Sajjadi, B. Schölkopf, and M. Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. *ICCV*, 2017. 4

[49] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey. Complex wavelet structural similarity: A new image similarity index. *TIP*, 2009. 2, 7, 14

[50] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002. 4, 5

[51] H. R. Sheikh, M. F. Sabir, and A. C. Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. *TIP*, 2006. 3

[52] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014. 1, 2, 5, 7, 12

[53] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang. Deep video deblurring for hand-held cameras. In *CVPR*, 2017. 4

[54] H. Talebi and P. Milanfar. Learned perceptual image enhancement. *ICCP*, 2018. 3

[55] H. Talebi and P. Milanfar. Nima: Neural image assessment. *TIP*, 2018. 3

[56] A. Tversky. Features of similarity. *Psychological review*, 1977. 2

[57] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 7

[58] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004. 1, 2, 8, 12, 14

[59] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *ICCV*, 2015. 4

[60] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *Signals, Systems and Computers*. IEEE, 2004. 1

[61] D. L. Yamins and J. J. DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 2016. 5, 8

[62] L. Zhang, L. Zhang, X. Mou, and D. Zhang. Fsim: A feature similarity index for image quality assessment. *TIP*, 2011. 1, 2, 12, 14

[63] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *ECCV*, 2016. 4, 5, 7

[64] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017. 1, 2, 5, 7, 12

Appendix

We show full quantitative details in Appendix A. We also discuss training details in Appendix B. Finally, we show results on the TID2013 dataset [45] in Appendix C.

A. Quantitative Results

In Table 5, we show full quantitative results across all validation sets and considered metrics, including low-level metrics, along with random, unsupervised, self-supervised, supervised, and perceptually-learned networks.

In Figures 7, 8, 9, we plot performance in individual validation sets. Figure 7 shows our traditional and CNN-based distortions, and Figures 8, 9 show results on real algorithm applications individually.

Human performance If humans chose patches $\{x_1,x_0\}$ with fraction $\{p,1-p\}$, the theoretical maximum for an oracle is $\max(p, 1-p)$. However, human performance is lower. If an agent chooses them with probability $\{q,1-q\}$, the agent will agree with $qp + (1-q)(1-p)$ humans on expectation. With a human agent, $q = p$, the expected human score is $p^2 + (1-p)^2$.

Linearly calibrating networks Learning linear weights on top of the ***Alex*** model achieves state-of-the-art results on the real algorithms test set. The ***linear*** models have a learned linear layer on top of each channel, whereas the out-of-the-box versions weight each channel equally. In Figure 10b, we show the learned weights for the ***Alex –frozen*** model. The `conv1-5` layers contain 64, 192, 384, 256, and 256 channels, respectively, for a total of 1152 weights. For each layer, `conv1-5`, 79.7%, 71.4%, 56.8%, 46.5%, 27.7%, respectively, of the weights are zero. This means that a majority of the `conv1` and `conv2` units are ignored,

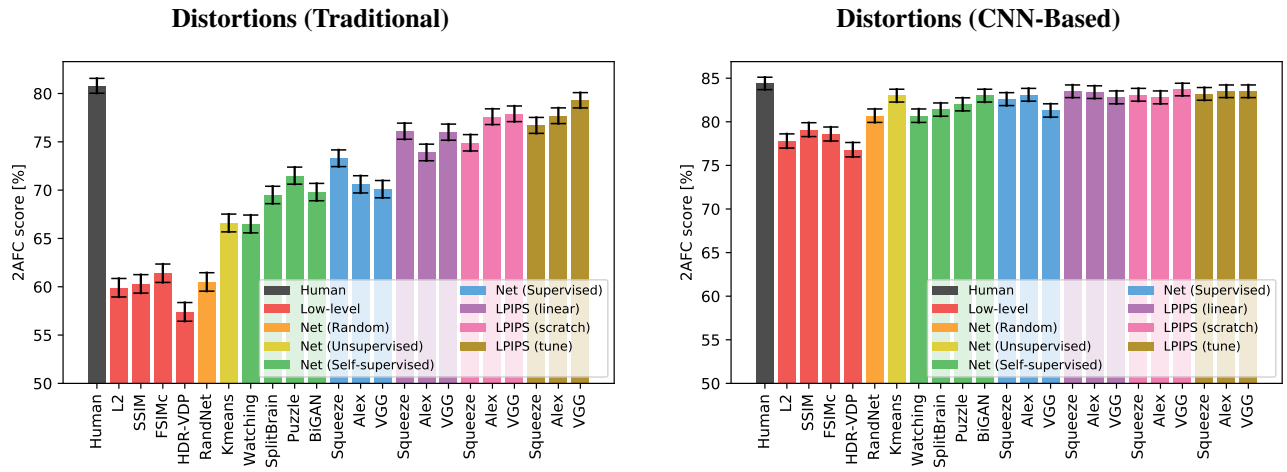


Figure 7: **Individual results** (left) traditional distortions (right) CNN-based distortions

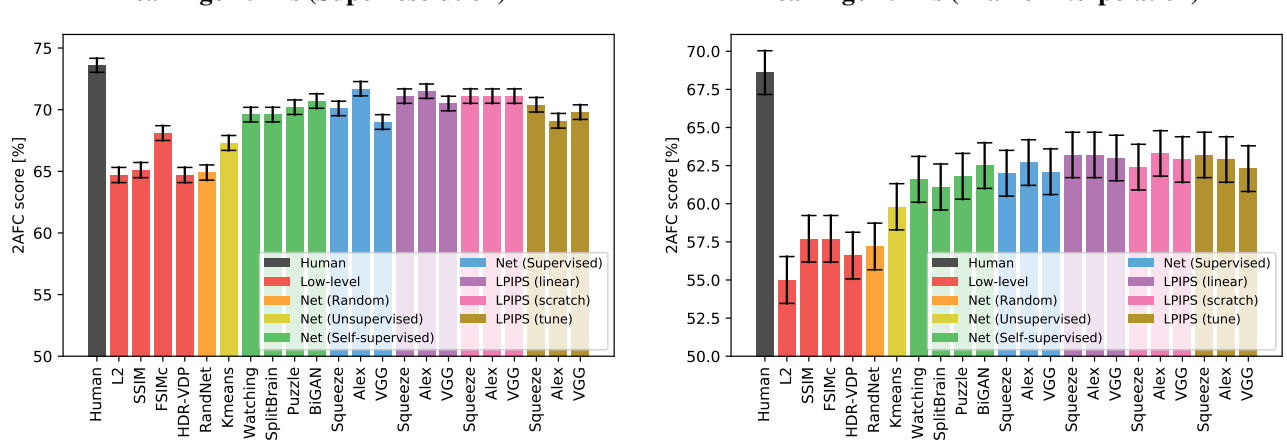


Figure 8: **Individual results** (left) superresolution (right) frame interpolation

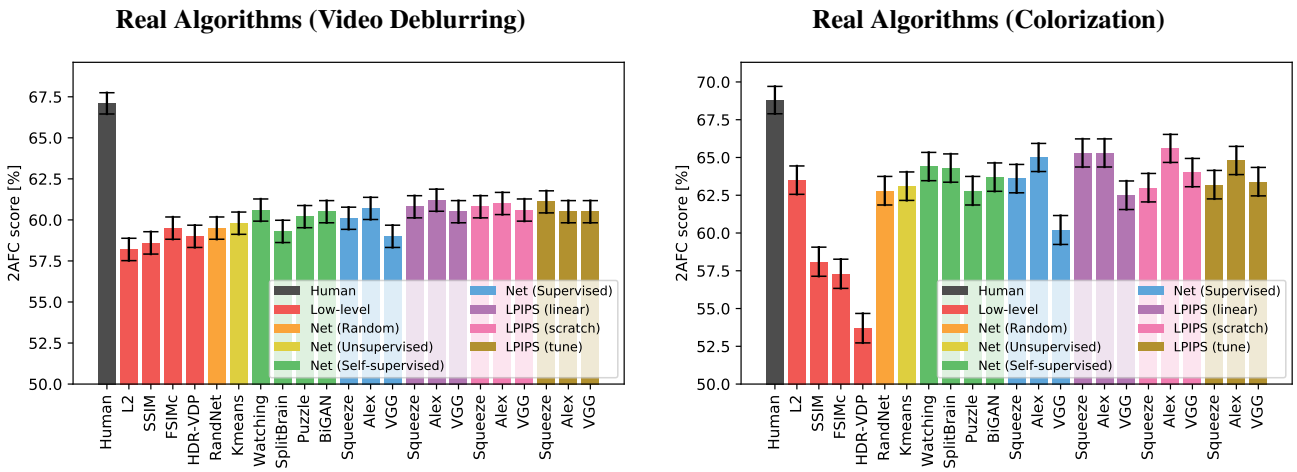


Figure 9: **Individual results** (left) video deblurring (right) colorization

Subtype	Metric	Distortions			Real Algorithms					All
		Traditional	CNN-Based	All	Super-res	Video Deblur	Colorization	Frame Interp	All	All
Oracle	Human	80.8	84.4	82.6	73.4	67.1	68.8	68.6	69.5	73.9
Low-level	L2	59.9	77.8	68.9	64.7	58.2	63.5	55.0	60.3	63.2
	SSIM [58]	60.3	79.1	69.7	65.1	58.6	58.1	57.7	59.8	63.1
	FSIMc [62]	61.4	78.6	70.0	68.1	59.5	57.3	57.7	60.6	63.8
	HDR-VDP [34]	57.4	76.8	67.1	64.7	59.0	53.7	56.6	58.5	61.4
Net (Random)	Gaussian	60.5	80.7	70.6	64.9	59.5	62.8	57.2	61.1	64.3
Net (Unsupervised)	K-means [26]	66.6	***83.0***	74.8	67.3	59.8	63.1	59.8	62.5	66.6
Net (Self-supervised)	Watching [43]	66.5	80.7	73.6	69.6	60.6	64.4	61.6	64.1	67.2
	Split-Brain [64]	69.5	81.4	75.5	69.6	59.3	64.3	61.1	63.6	67.5
	Puzzle [40]	71.5	82.0	76.8	70.2	60.2	62.8	61.8	63.8	68.1
	BiGAN [13]	69.8	***83.0***	76.4	70.7	60.5	63.7	62.5	64.4	***68.4***
Net (Supervised)	SqueezeNet [20]	**73.3**	***82.6***	**78.0**	70.1	60.1	63.6	62.0	64.0	***68.6***
	AlexNet [27]	70.6	**83.1**	76.8	**71.7**	60.7	65.0	62.7	***65.0***	**68.9**
	VGG [52]	70.1	81.3	75.7	69.0	59.0	60.2	62.1	62.6	67.0
*LPIPS (Learned Perceptual Image Patch Similarity)	Squeeze – lin	76.1	83.5	79.8	71.1	***60.8***	***65.3***	***63.2***	***65.1***	70.0
	Alex – lin	73.9	83.4	78.7	***71.5***	**61.2**	***65.3***	***63.2***	**65.3**	69.8
	VGG – lin	76.0	82.8	79.4	70.5	60.5	62.5	***63.0***	64.1	69.2
	Squeeze – scratch	74.9	83.1	79.0	71.1	***60.8***	63.0	62.4	64.3	69.2
	Alex – scratch	77.6	82.8	80.2	71.1	***61.0***	**65.6**	**63.3**	***65.2***	**70.2**
	VGG – scratch	77.9	**83.7**	80.8	71.1	60.6	64.0	***62.9***	64.6	70.0
	Squeeze – tune	76.7	83.2	79.9	70.4	***61.1***	63.2	***63.2***	64.5	69.6
	Alex – tune	77.7	83.5	80.6	69.1	60.5	64.8	***62.9***	64.3	69.7
	VGG – tune	**79.3**	83.5	**81.4**	69.8	60.5	63.4	62.3	64.0	69.8

Table 5: **Results**. We show 2AFC scores (higher is better) across a spectrum of methods and test sets. The **bolded & underlined** values are the highest performing. The ***bolded & italicized*** values are within 0.5% of highest. *LPIPS metrics are trained on the same traditional and CNN-based distortions, and as such have an advantage relative to other methods when testing on those same distortion types, even on unseen test images. These values are indicated by gray values. The best gray value per column is also **bolded**.

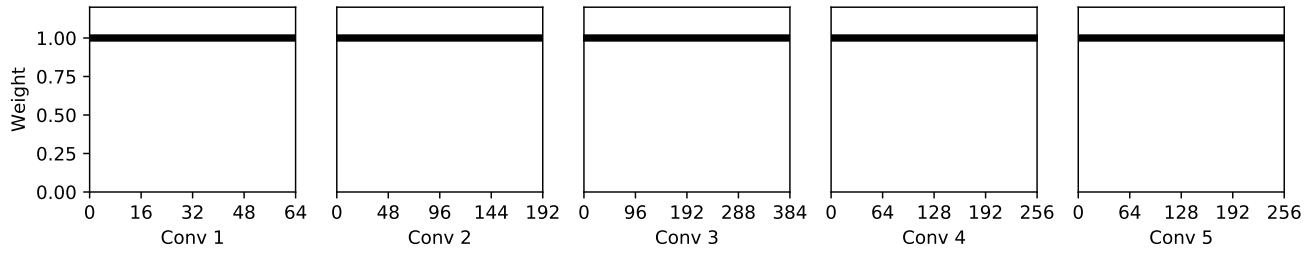
and almost all of the `conv5` units are used. Overall, about half of the units are ignored. Taking the cosine distance is equivalent to setting all weights to 1 (Figure 10a).

Data quantity for training models on distortions The performance of the validation set on our distortions (80.6% and 81.4% for ***Alex – tune*** and ***VGG – tune***, respectively), is almost equal to human performance of 82.6%. This indicates that our training set size of 150k patch pairs and 300k judgments is nearly large enough to fully explore the traditional and CNN-based distortions which we defined. However, there is a small gap between the ***tune*** and ***scratch*** models (0.4% and 0.6% for ***Alex*** and ***VGG***, respectively).

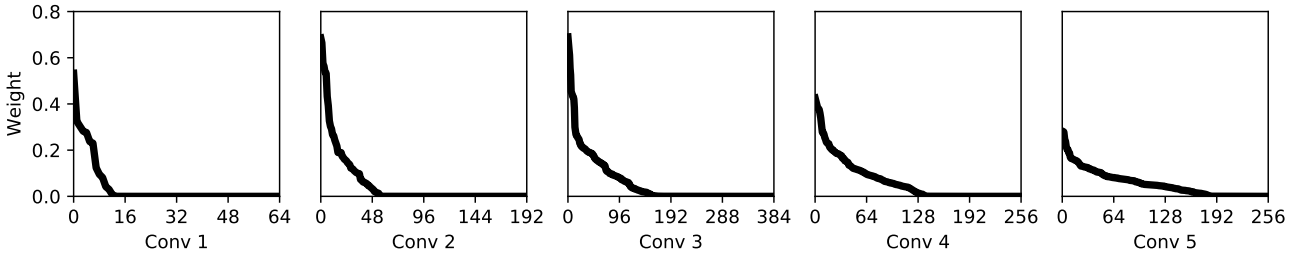
B. Model Training Details

We illustrate the loss function for training the network in Figure 3 (right) and describe it further in the supplementary material. Given two distances, (d_0, d_1), we train a small network $\mathcal{G}$ on top to map to a score $\hat{h} \in (0, 1)$. The architecture uses two 32-channel `FC-ReLU` layers, followed by a 1-channel `FC` layer and a sigmoid. Our final loss function is shown in Equation 2.

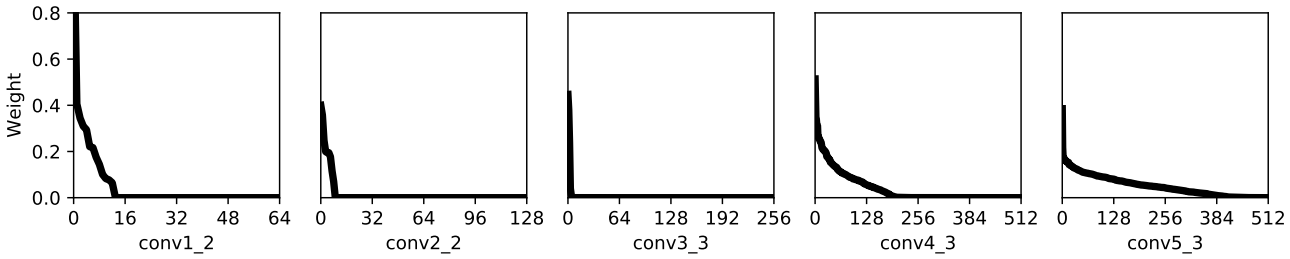
$$\begin{aligned}\mathcal{L}(x, x_0, x_1, h) = -h \log \mathcal{G}(d(x, x_0), d(x, x_1)) \\ -(1-h) \log(1 - \mathcal{G}(d(x, x_0), d(x, x_1)))\end{aligned} \tag{2}$$



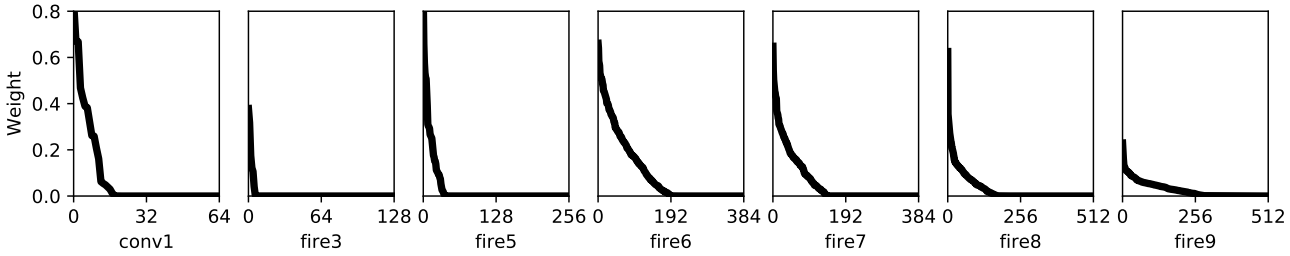
(a) **Unlearned weights for AlexNet model (cosine distance)**



(b) **Learned weights from *Alex–lin* model**



(c) **Learned weights from *VGG–lin* model**



(d) **Learned weights from *Squeeze–lin* model**

Figure 10: **Learned linear weights by layer.** (a) Unlearned weights correspond to using weighting 1 for each channel in each layer, which results in computing cosine distance. (b) We show the learned weights from each layer of our *Alex–lin* model. This is the w term in Figure 3. Each subplot shows the channel weights from each layer, sorted in descending order. The x-axis shows the channel number, and y-axis shows the weight. Weights are restricted to be non-negative, as image patches should not have negative distance. (c,d) Same as (b), but with the *VGG–lin* and *Squeeze–lin* models.

In preliminary experiments, we also tried a ranking loss, which attempts to force a constant margin between patch pairs $d(x, x_0)$ and $d(x, x_1)$. We found that using a learned network, rather than enforcing the same margin in all cases, worked better.

Here, we provide some additional details on model training for our networks trained on distortions. We train with 5 epochs at initial learning rate 10^{-4}, 5 epochs with linear decay, and batch size 50. Each training patch pair is judged 2 times, and the judgments are grouped together. If, for example, the two judges are split, then the classification target (h in Figure 3) will be set at 0.5. We enforce non-negative weightings on the linear layer w, since larger distances in a certain feature should not result in two patches becoming closer in the distance metric. This is done by projecting the weights into the constraint set at every iteration. In other words, we check for any negative weights, and force them to be 0. The project was implemented using `PyTorch` [42].

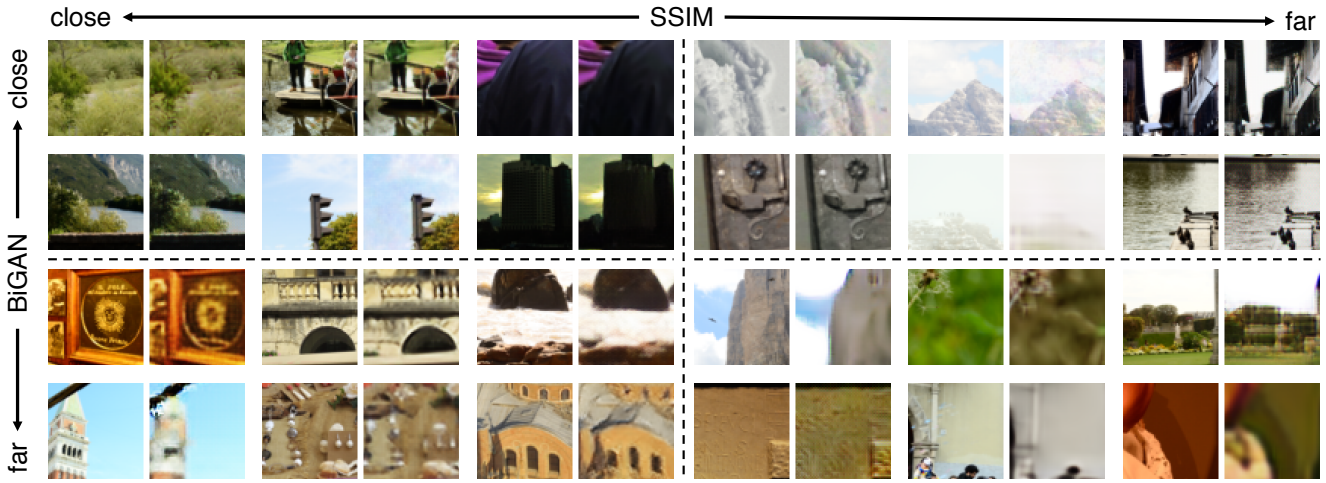


Figure 11: **Qualitative comparisons on distortions.** We show qualitative comparison on CNN-based distortions, using the SSIM [58] metric and BiGAN network [13]. We show examples where both agree the patches are closer or far, and examples where the metrics disagree. A primary difference is that deep embeddings appear to be more sensitive to blur. Please see the appendix for additional examples.

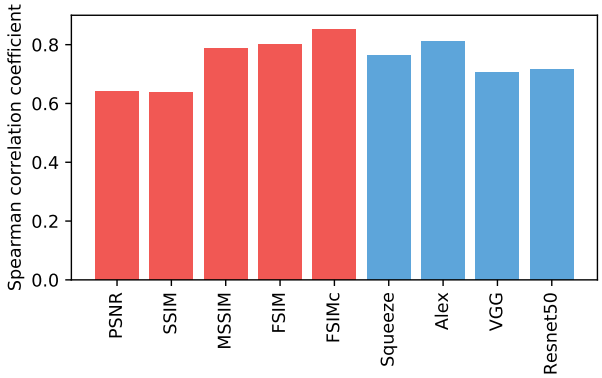


Figure 12: **TID Dataset** We show the Spearman correlation coefficient of various methods on the TID2013 Dataset [45]. Note that deep networks trained for classification perform well out of the box (blue).

C. TID2013 Dataset

In Figure 12, we compute scores on the TID2013 [45] dataset. We test the images at a different resolutions, using $\{128, 192, 256, 384, 512\}$ for the smaller dimension. We note that even averaging across all scales and layers, with no further calibration, the AlexNet [27] architecture gives scores near the highest metric, FSIMc [62]. On our traditional perturbations, the FSIMc metric achieves 61.4%, close to ℓ_2 at 59.9%, while the deep classification networks we tested achieved 73.3%, 70.6%, and 70.1%, respectively. The difference is likely due to the inclusion of geometric distortions in our dataset. Despite their frequent use in such situations, metrics such as SSIM were not designed to handle geometric distortions [49].

D. Changelog

v1 initial preprint release

v2 CVPR camera ready; moved TID results (Appendix C), SSIM vs BiGAN (Figure 11), and some training details into the Appendix to fit into 8 page limit; clarified that SSIM was not designed to handle geometric distortions [49] and clarified that our dataset is a perceptual similarity dataset (as opposed to an IQA dataset); added linear weights for *Squeeze-lin* and *VGG-lin* architectures in Figure 10; miscellaneous small edits to text.