The Unreasonable Effectiveness of Deep Features as a Perceptual Metric

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

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 $\ell_2$ 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 $\ell_2$ 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 [57], MSSIM [59], FSIM [61], and HDR-VDP [34].

However, constructing a perceptual metric is challenging, because human judgments of similarity (1) depend on high-order image structure [57], (2) are context-dependent...
self-supervised networks, including BiGANs [13], cross-
monly used metrics like SSIM and FSIM [57, 61], which
correspond far better than the comparisons 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 [51] 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, 51] 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 [57, 61], which were not designed to handle situations where spatial ambiguities are a factor [48]. Furthermore, the best performing self-supervised networks, including BiGANs [13], cross-
channel prediction [63], 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 [44]. 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 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.

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 [50], TID2008 [45], CSIQ [29], and TID2013 [44] 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 [53, 54] 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 -
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.

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

Table 2: **Parameter type**

<table>
<thead>
<tr>
<th>Input corruption</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>null, pink noise, white noise, color removal, downsampling</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generator network architecture</th>
<th># layers with dropout, force skip connection at highest layer, upsampling method, normalization method, first layer stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discriminator</td>
<td>number of layers</td>
</tr>
<tr>
<td>Loss/Learning</td>
<td>weighting on pixel-wise ($\ell_1$), VGG, discriminator losses, learning rate</td>
</tr>
</tbody>
</table>

*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 [46] 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, 58, 31, 47]. A common qualitative way of presenting superresolution results is zooming into specific patches and comparing differences. As such, we sample random $64 \times 64$ 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 Middlesbury dataset [49]. 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 [52], along with deblurring outputs from Photoshop Shake Reduction, Weighted Fourier Aggregation [11], and three variants of a deep video deblurring method [52].

*Colorization* We sample patches using random scales on the colorization task, on images from the ImageNet dataset [46]. The algorithms are from pix2pix [22], Larsen et al. [30], and variants from Zhang et al. [62].

### 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 $T$ 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 [44], has 500k judgments on 3000 distortions (from 25 input images with 24 distortions types, 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Data source</th>
<th>Train/Val</th>
<th># Examples</th>
<th># Judge /Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional</td>
<td>[7]</td>
<td>Train</td>
<td>56.6k</td>
<td>2</td>
</tr>
<tr>
<td>CNN-based</td>
<td>[7]</td>
<td>Train</td>
<td>38.1k</td>
<td>2</td>
</tr>
<tr>
<td>Mixed</td>
<td>[7]</td>
<td>Train</td>
<td>56.6k</td>
<td>2</td>
</tr>
<tr>
<td>2AFC–Distort [Trn]</td>
<td>–</td>
<td>Train</td>
<td>151.4k</td>
<td>2</td>
</tr>
<tr>
<td>Traditional</td>
<td>[10]</td>
<td>Train</td>
<td>4.7k</td>
<td>5</td>
</tr>
<tr>
<td>CNN-based</td>
<td>[10]</td>
<td>Train</td>
<td>4.7k</td>
<td>5</td>
</tr>
<tr>
<td>2AFC–Distort [Val]</td>
<td>–</td>
<td>Val</td>
<td>9.4k</td>
<td>5</td>
</tr>
<tr>
<td>Superres</td>
<td>[32]</td>
<td>Val</td>
<td>10.9k</td>
<td>5</td>
</tr>
<tr>
<td>Frame Interp</td>
<td>[49]</td>
<td>Val</td>
<td>1.9</td>
<td>5</td>
</tr>
<tr>
<td>Colorization</td>
<td>[46]</td>
<td>Val</td>
<td>4.7</td>
<td>5</td>
</tr>
<tr>
<td>2AFC–Real Alg [Val]</td>
<td>–</td>
<td>Val</td>
<td>26.9k</td>
<td>5</td>
</tr>
<tr>
<td>Traditional</td>
<td>[10]</td>
<td>Val</td>
<td>4.8k</td>
<td>3</td>
</tr>
<tr>
<td>CNN-based</td>
<td>[10]</td>
<td>Val</td>
<td>4.8k</td>
<td>3</td>
</tr>
<tr>
<td>JND–Distort</td>
<td>–</td>
<td>–</td>
<td>9.6k</td>
<td>3</td>
</tr>
</tbody>
</table>

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.

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).

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 [51] 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 [60]. 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 subsequent “fire” modules.

We additionally evaluate self-supervised methods, including puzzle-solving [40], cross-channel prediction [62, 63], learning from video [42], and generative modeling [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 \(\hat{y}_l^i, \hat{y}_0^i \in \mathbb{R}^{H_l \times W_l \times C_l}\) for layer \(l\). We scale the activations channel-wise by vector \(w^l \in \mathbb{R}^{C_l}\) and compute the \(\ell_2\) distance. Finally, we average spatially and sum channel-wise. Note that using \(w_l = 1V_l\) is equivalent to computing cosine distance.

\[
d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} ||w_l \odot (\hat{y}_l^h w^l - \hat{y}_0^h w^l)||_2^2 \quad (1)
\]

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.
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 [63], and segmenting foreground objects from video [42]. 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.

In Table 5, we explore how well our perceptual task correlates to semantic tasks on the PASCAL dataset [15], using results summarized in [63], including additional self-supervised methods [1, 43, 12, 56, 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?**

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.

**Does training on traditional and CNN-based distortions need classification?**

Does the network have to be trained on classification?
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 6, we show a qualitative comparison across our traditional distortions for a deep method, BiGANs [13], and a representation traditional perceptual method, SSIM [57]. 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 [60]. 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

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