

Supplemental Material for Evaluating ‘Graphical Perception’ with CNNs

Daniel Haehn, James Tompkin, and Hanspeter Pfister

1 ADDITIONAL EXPERIMENTS

1.1 Anti-aliasing

One way to increase the fidelity of the information given to the network for angled or curved lines is to anti-alias the line drawing. This provides grayscale intensity values to smooth out the jagged line edges, and may be more suitable to networks trained on natural images (such as VGG19* and Xception*, with ImageNet weights).

We conduct an experiment to measure continuous pie chart angles from stimuli with anti-aliased (AA) lines (Figure 1). There is a slight increase in performance for the AA case for VGG19*, and an even slighter increase for Xception*. However, overall, adding anti-aliasing to the line generation was not statistically significant to the tested CNNs when comparing the performance across all networks ($F(1, 30) = 0.341, p > 0.5$). This is not surprising since smoothing only changes very little information at the resolution of our stimuli. Smooth edges might add more value on higher resolution stimuli which we plan to evaluate in the future.

1.2 Noise

For all our experiments, we add subtle 5% noise to every pixel to enhance variability. We did not observe a significant effect on regression performance when comparing the Weber-Fechner’s law experiment with and without noise, averaged over 4 runs (Figure 2). However, the variability of LeNet increased with the additional complexity.

2 COMPARISONS TO CLEVELAND AND MCGILL’S STIMULI

We visually compare our stimuli to those of Cleveland and McGill [1], and explain our choices for any differences.

2.1 Resolution Differences

Cleveland and McGill’s stimuli were created on 8.5×11 inch paper and presented to participants in a binder. While it is unclear from their paper how the stimuli were created (e.g., printed from a digital file, or drawn with pen), it is safe to assume that there is a resolution difference from our stimuli. Our stimuli are 100×100 digital images, with their resolution chosen for computational reasons: We needed to train more than 2,500 models and, at this resolution, our most complex Xception model took 6 hours to train with 60,000 stimuli.

This limited resolution affects network performance in making precise estimates, especially in tasks which require estimating lines drawn at angles or curved lines. As we draw binary stimuli, there are only so many angles of lines that can be drawn within 100×100 pixels that produce different visual output. This leads to a performance floor or constant error. For instance, on average, we can only draw binary angled lines with 1.45 degrees of accuracy (0.67 standard deviation).

2.2 Elementary Perceptual Tasks

Cleveland and McGill provide an explanatory figure for their defined elementary perceptual tasks, with each example showing the expected variation in parameter through two samples. We compare visually

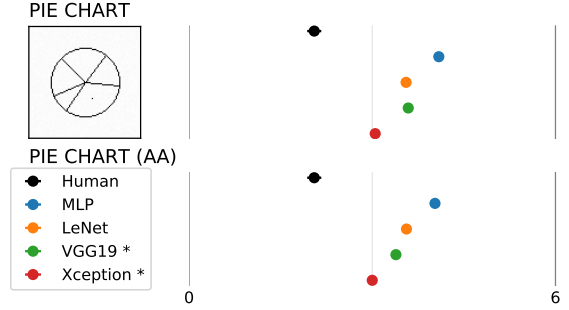


Fig. 1: **Anti-aliasing.** We test whether anti-aliasing affects the performance of our networks on pie charts by measuring MLAE. Starred networks are fine tuned from ImageNet weights. The difference is not statistically significant ($F(1, 30) = 0.341, p > 0.5$).

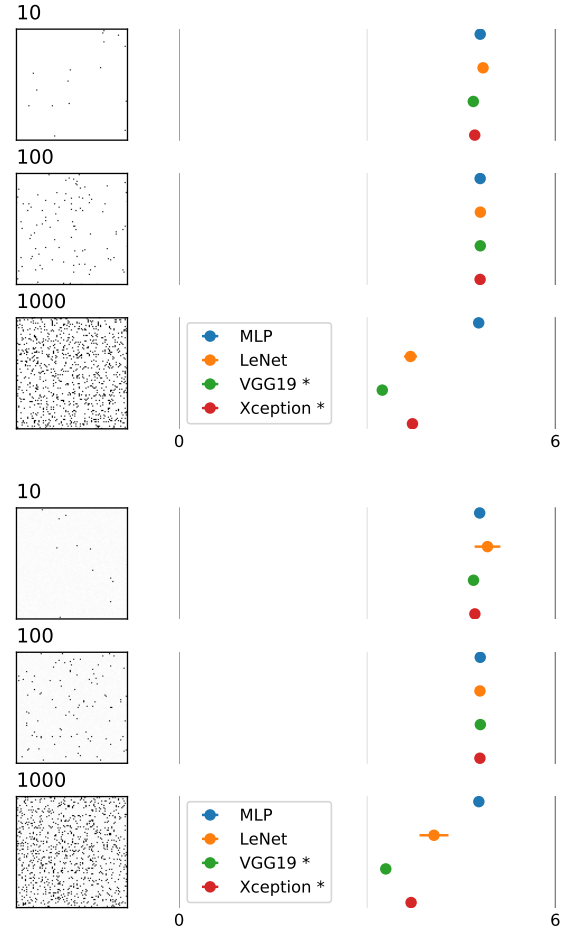


Fig. 2: **Noise.** We test whether noise affects the performance of our networks on the Weber-Fechner’s law experiment by measuring MLAE. *Top:* With noise, $MLAE = 4.511$ ($SD = 0.512$). *Bottom:* Without noise $MLAE = 4.491$ ($SD = 0.543$). The difference is not statistically significant ($F(1, 22) = 0.008, p > 0.5$).

- Daniel Haehn, and Hanspeter Pfister are with Harvard University.
E-mail: {haehn,pfister}@seas.harvard.edu.
- James Tompkin is with Brown University.
E-mail: james_tompkin@brown.edu.

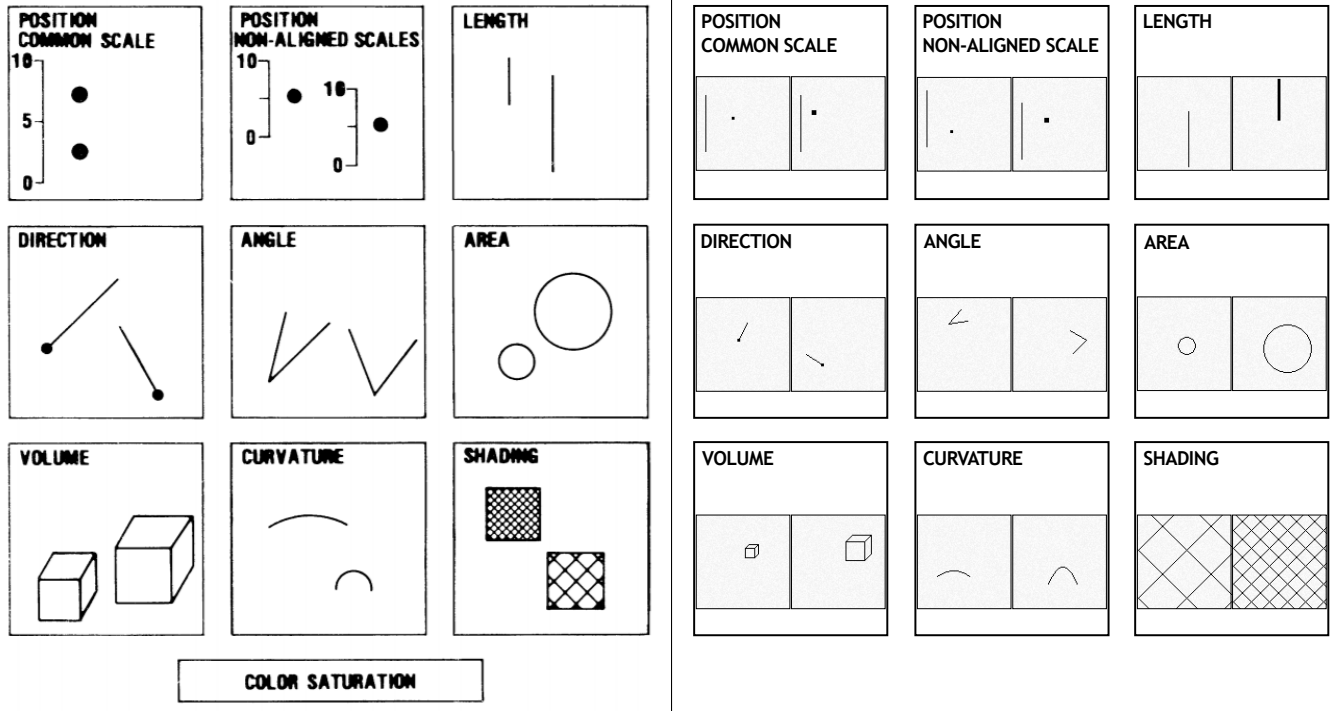


Fig. 3: *Left*: Cleveland and McGill’s example figure for their elementary perceptual tasks, with each task showing variation through two sample examples (Figure 1 from [1]). *Right*: Two example stimuli from our elementary perceptual task experiment. Note that the border around each of our stimuli only exist in this figure to visually separate the stimuli; no border exists in our experiments. Our stimuli also include minor variations in style, such as dot and stroke width in position and length. Finally, as per Cleveland and McGill, we defer color saturation as a perceptual problem for future work.

to two samples of our stimuli (Figure 3). Our stimuli are similar in appearance, with the addition that our stimuli also include minor variations in style such as dot and stroke width in position and length.

One difference is that we exclude the numerical values written as text along the scale in the ‘Position Common Scale’ and ‘Position Non-aligned Scales’ examples. While training a CNN to recognize text on graphs is possible given enough examples [3], in our case these numbers would not change from stimuli to stimuli and so the network would not gain any information between stimuli to help solve the position estimation problem. As such, we simplify the task by removing the redundant numbers, with the scale values implicitly encoded by the size of the image and the location of the line start and end pixels.

By the same reasoning, the scale itself is redundant in the ‘Position Common Scale’ setting as it does not move—the network must map the vertical pixel positions of the dots to the quantity range of numbers. We leave the scale in to allow comparison with the ‘Position Non-aligned Scale’ task. Further, when we train networks which look at *all* elementary perceptual tasks at once, leaving in the scale allows the network to exploit the consistency of representation across the two position estimation tasks.

Further, we list the permutations of each stimuli in Table 1.

2.3 Position-length Experiment

Again, we attempted to replicate Cleveland and McGill’s stimuli as closely as possible, while removing the text on the scale and the A/B category labels (Figure 4). These values are implicitly encoded by the vertical height within the image, and by the left/right position of the bars. Similar to Cleveland and McGill, we place a dot within the regions which require ratio estimation. In our case, the network must learn the association of dot placement to required length estimation.

2.4 Position-angle Experiment

As before, our figures make implicit the scales and labels within these stimuli (Figure 5). One difference here is that Cleveland and McGill ex-

PLICITLY tell the participants on a supplemental sheet which pie segment or bar is largest; we replace this signal with a visual dot.

2.5 Bars/Rectangles Experiment

In this case, the only difference is that we remove the A/B category labels, and let the comparison judgment be implicitly encoded by the left/right bars (Figure 6).

2.6 Point Cloud Experiment

Cleveland and McGill have no equivalent to this Weber-Fechner’s law experiment, and so we show no stimuli comparison.

3 COMPARING THE HUMAN VS. MACHINE JUDGMENT PROCESSES

We describe at a high level what happens when a human is asked to predict from the ‘position common scale’ stimuli, and what happens when a CNN is asked to predict given the stimuli as input.

3.1 Human Judgment

For training, a human being learns how to see through experience gained by existing in the world, such as our ability to interpret visual edges as lines and to read numbers. Given this experience, upon observing the ‘position common scale’ stimuli at prediction time:

1. The human looks at the scale and reads the demarcation ticks and text labels. This lets the human know the numerical values associated with the visual marks, and defines the scale bounds.
2. The human looks at the the dot and mentally projects a horizontal line over to intersect the scale.
3. The human estimates the distance between the two closest demarcation ticks and the intersection line. The human may make additional mental ticks within the span to help in the estimation.
4. The human converts this into the scale using knowledge of the tick values, producing a prediction.

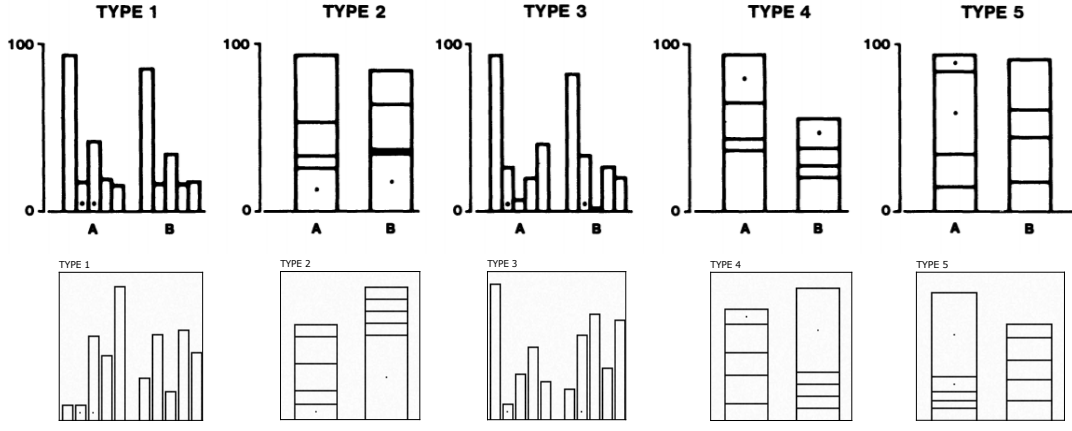


Fig. 4: *Top*: Cleveland and McGill's example stimuli for their position-length experiment (Figure 4 from [1]). *Bottom*: Example stimuli from our position-length experiment. Note that the border around each of our stimuli only exist in this figure; no border exists in our experiments.

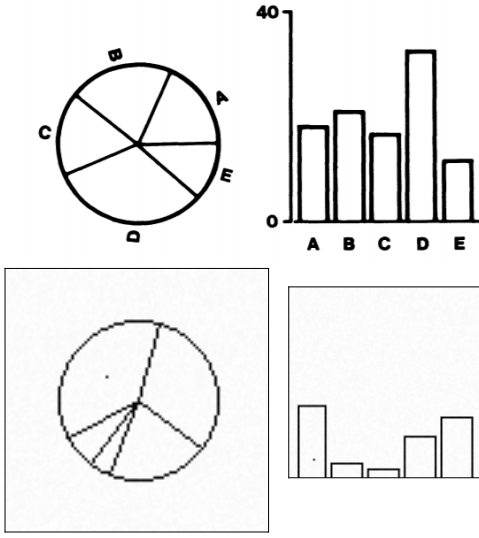


Fig. 5: *Top*: Cleveland and McGill's example stimuli for their position-angle experiment (Figure 3 from [1]). *Right*: Example stimuli from our position-length experiment. Note that the border around each of our stimuli only exist in this figure; no border exists in our experiments. Further, we have resized our stimuli on the page to more clearly show the similarity; each are 100×100 pixels.

The first two steps may be interchanged, as the output of each is stored in human memory in either case.

3.2 Brief CNN Introduction

For a CNN, the process is harder to describe in intuitive terms, and so we provide a brief introduction to how CNNs work. The network is defined to have a set of layers, with each layer feeding into the next, and each typically performing some subset of three operations: convolution with kernels, non-linear transformation with activation functions, and pooling. Following these, a densely-connected set of neurons known as a multi-layer perceptron (MLP) learns how to combine the convolution layer outputs to make a prediction.

Trainable Convolution. The network has a number of kernels within each layer, e.g., in layer one of LeNet, we have 20 trainable 5×5 kernels. We convolve an input image with our kernels to produce a 'feature map' measuring the kernel response to the image. Once trained, these kernels identify important visual elements in a scene, similar to a template matching process. In a loose biological comparison, kernels in

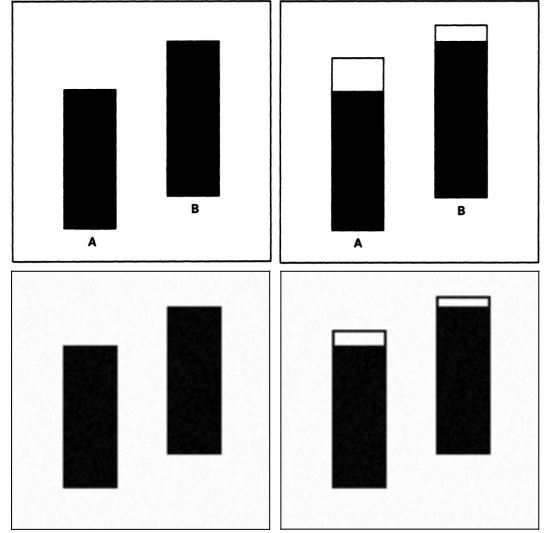











Fig. 6: *Top*: Cleveland and McGill's example bars and rectangles (Figure 12 from [1]). *Right*: Example stimuli from our bars and rectangles experiment. Note that the border around each of our stimuli only exist in this figure; no border exists in our experiments.

early layers are often compared to functions in the early human visual system which identify contrast or edges, which is an important task within graphical perception. Through hierarchical combination, kernels in later layers are said to localize and identify objects (if the network is trained on natural images) [4]. In our graphical perception setting, this would potentially let a network identify and compare larger structures like the cube in our volume task, or glyphs.

Activation Functions. With the feature maps computed by the convolution with trained kernels, we apply the activation function, which defines how much we wish our network to respond to a given feature. LeNet uses the Rectified Linear Unit (ReLU) activation function, which sets all values less than 0 to 0, and is loosely said to be biologically inspired by one-sided human neuron activations. ReLU helps us train deeper neural networks by propagating large positive signals [2].

Pooling. A pooling layer acts to summarize a response over a spatial region of the feature map, either by averaging or by selecting the highest response from the region ('max. pooling'). LeNet uses 2×2 max pooling, which decreases the size of the feature map by two in each dimension. Across layers, pooling lets the network accumulate or select activations as part of its prediction process.

Table 1: **Elementary Perceptual Tasks.** Rasterized visualizations of our elementary perceptual tasks as defined by Cleveland and McGill [1] (color saturation excluded). We sequentially increase the number of parameters for every task (e.g., by adding translation). This introduces variability and creates increasingly more complex datasets.

Elementary Perceptual Task	Permutations
 <i>Position Common Scale</i> Position Y + Position X + Spot Size	60 3,600 21,600
 <i>Position Non-Aligned Scale</i> Position Y + Position X + Spot Size	600 36,000 216,000
 <i>Length</i> Length + Position Y + Position X + Width	60 2,400 144,000 864,000
 <i>Direction</i> Angle + Position Y + Position X	360 21,600 1,296,000
 <i>Angle</i> Angle + Position Y + Position X	90 5,400 324,000
 <i>Area</i> Radius + Position Y + Position X	40 800 16,000
 <i>Volume</i> Cube Sidelength + Position Y + Position X	20 400 8,000
 <i>Curvature</i> Midpoint Curvature + Position Y + Position X	80 1,600 64,000
 <i>Shading</i> Density + Position Y + Position X	100 2,000 40,000

MLP. Finally, after a series of layers (two for LeNet: Conv→ReLU→MaxPool→Conv→ReLU→MaxPool), we feed the resulting downsampled feature map into a multi-layer perceptron, which is connected to every pixel in the feature map. Each perceptron in the MLP is a linear classifier which is trained to weight the feature maps to produce the correct prediction. The MLP itself has two layers separated by a ReLU to allow for complex combination.

3.3 CNN Judgment.

Now, let us predict for the ‘position common scale’ task. Through training, our network has learned a set of convolutional kernels across its layers which, in combination, help the network to predict estimates for unseen test data.

1. The kernels produce strong responses in parts of the test image that were helpful to predict the training data. These are typically edges or visualization parts such as bars. For ‘position common scale’, early-layer kernels produce a strong response for dots.
2. Pooling selects the most important features from across the image. Over a hierarchy of layers, these are trained to be combined by kernels into more useful features. For ‘position common scale’,

this dot response would be propagated through the layers, but the exact position of the dot would be made imprecise.

3. The MLP has been trained to weight (or combine) the pooled feature maps into a prediction. For ‘position common scale’, the MLP will weight the feature map response at different locations to map to a value between 0–60.

3.4 Human/CNN Comparison

While parts of this process are similar, such as the early visual system identifying edges, the high-level picture is that the human and CNN prediction processes are substantially different. The human prediction involves implicit geometric understanding of a space, with projection and line length comparisons. The CNN prediction has no explicit way to represent these high-level operations; instead, they must be implicitly represented through a series of low-level weighted sums of pixel responses, clamping activation function transformations, and spatially-local max operations.

However, for some perception tasks like area estimation or our point cloud experiment in the main paper, this series of operations is beneficial. For instance, estimating the number of points added to the point cloud can be computed exactly by summing over the stimuli and subtracting the base number of points (10, 100, 1,000). This is a task that is extremely tedious at best for a human, but easier for the CNN given its layer aggregation methodology.

4 SUPPLEMENTAL RESULTS

We present several plots which contain complete results for the elementary perceptual task experiment, for which, given the number of experiments, the main paper presented only a selection. We report MLAE for all added parameters to the stimuli, rather than just the most complex parameterization as in the main paper (Figures 7 and 8). From this, we see that performance is largely equal across parameters, showing that most networks have sufficient capacity for the given parameterizations.

We also report complete results for the cross-network variability experiment on the elementary perceptual task experiment (Figure 9). As in the main paper, this shows that our networks are not able to generalize to additional translation or stroke width parameters without representative training data.

Further, we show how the errors for each network are distributed, across elementary perceptual tasks and across different cross-validation splits (Figure 10). Most errors are approximately normally distributed, though our CNN with less parameters (LeNet) and our MLP often have errors which are farther from a normal distribution, and show structure.

REFERENCES

- [1] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554, 1984.
- [2] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In G. Gordon, D. Dunson, and M. Dudk, eds., *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15 of *Proceedings of Machine Learning Research*, pp. 315–323. PMLR, Fort Lauderdale, FL, USA, 11–13 Apr 2011.
- [3] K. Kafle, S. Cohen, B. Price, and C. Kanan. Dvqa: Understanding data visualizations via question answering. In *CVPR*, 2018.
- [4] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds., *Computer Vision – ECCV 2014*, pp. 818–833. Springer International Publishing, Cham, 2014.

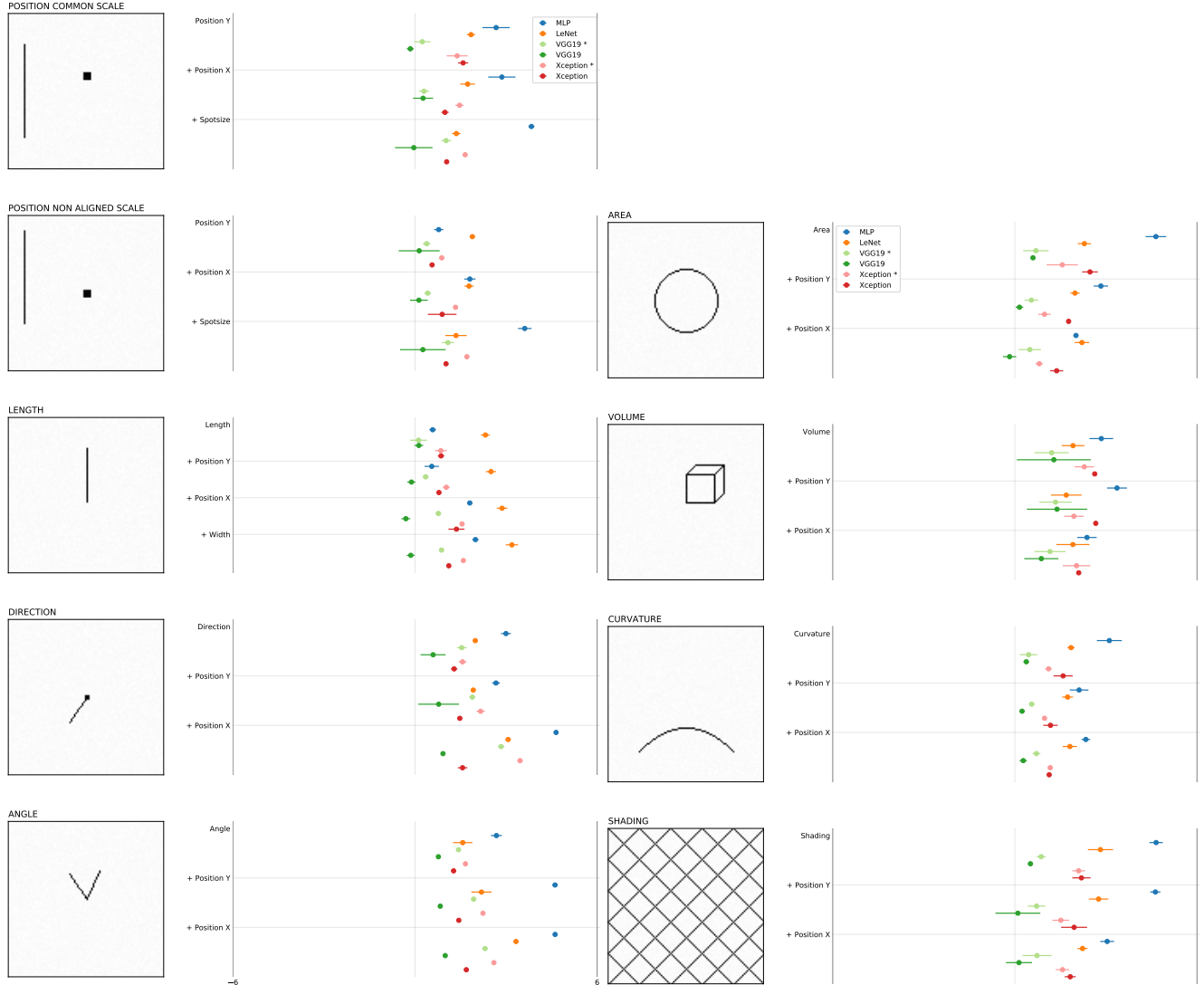


Fig. 7: **Elementary perceptual tasks.** Midmean logistic absolute errors (MLAE) for all generated stimuli and across all networks. The * indicates networks which use ImageNet weights instead of being trained from scratch.

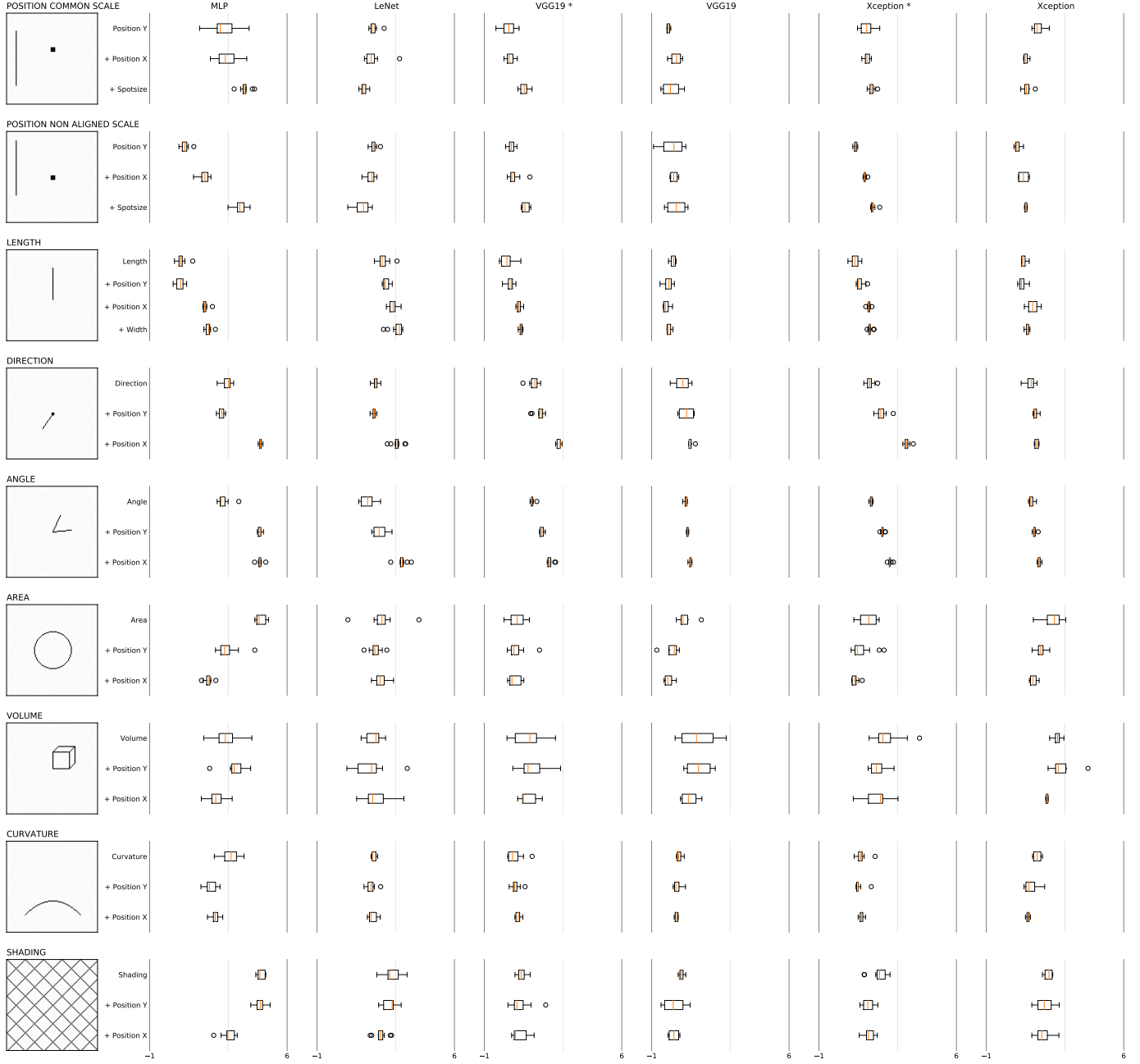


Fig. 8: Elementary perceptual tasks. Midmean logistic absolute errors (MLAE) visualized as box plots.



Fig. 9: **Cross-network variability.** Our networks fail when the stimuli changes through translation or stroke width. The x-labels indicate the training configuration while the y-labels indicate the stimuli variation. Numbers represent MLAE.

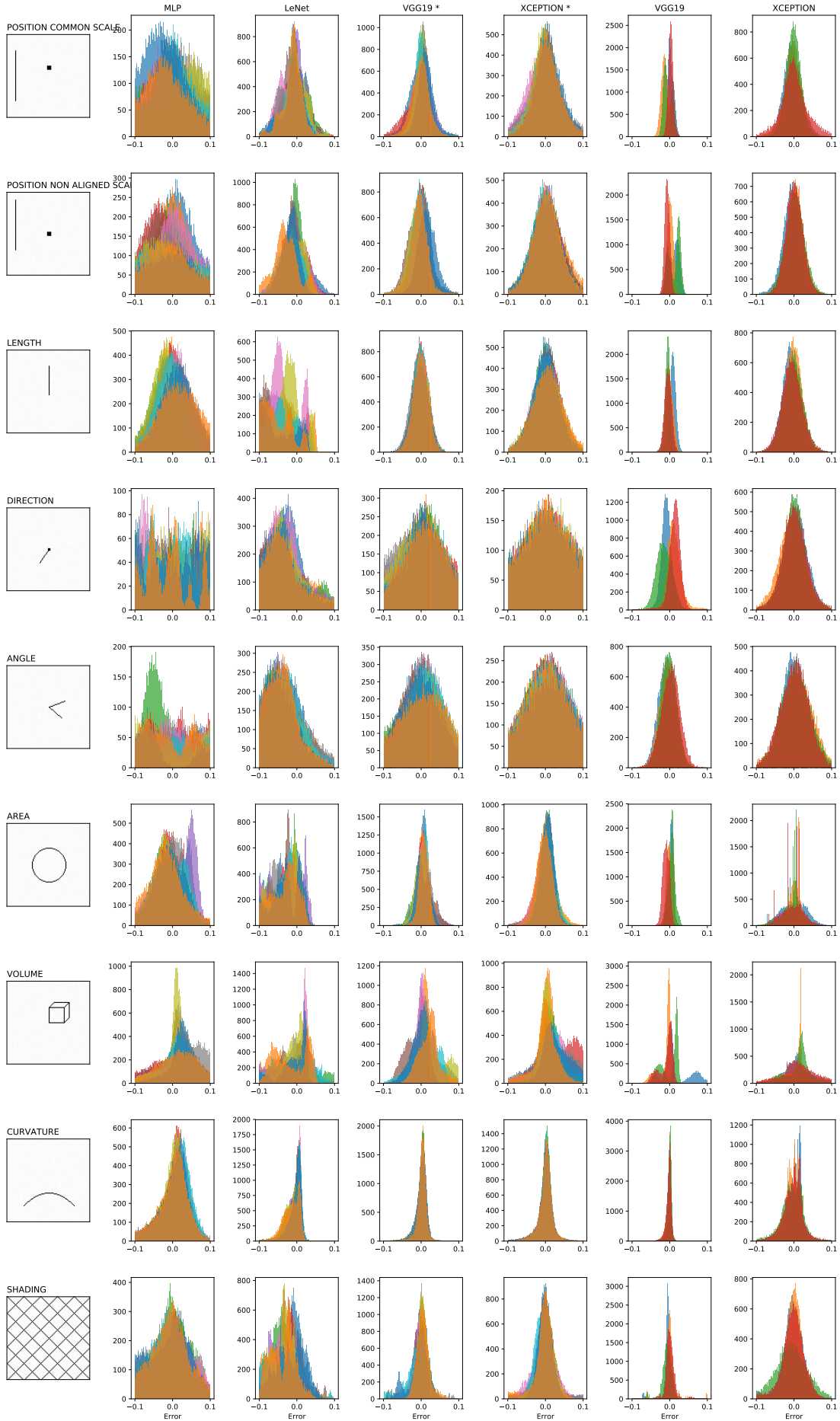


Fig. 10: **Error distributions.** Error distributions of our networks when decoding elementary perceptual tasks.

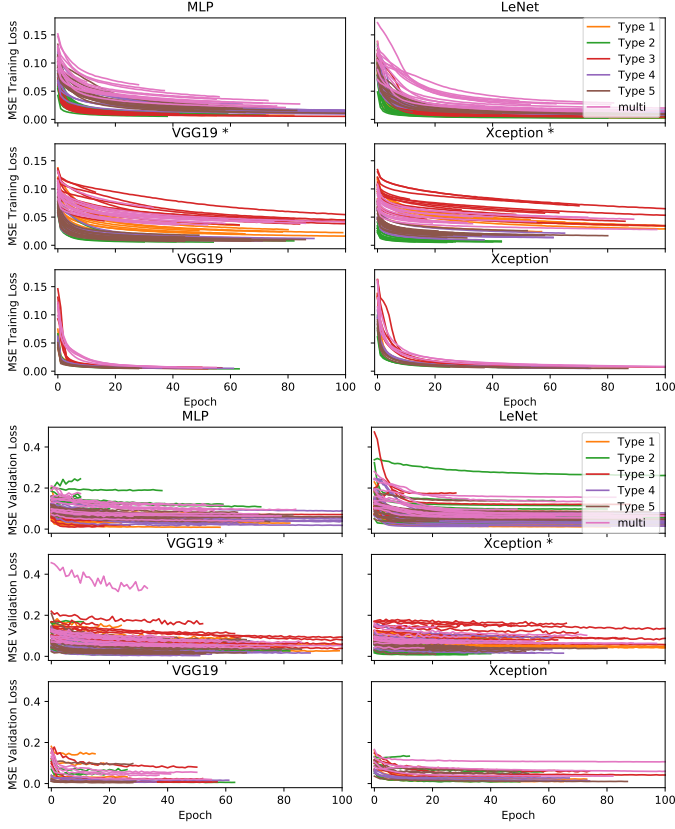


Fig. 11: **Loss plots for the position-length experiment.** We visualize the MSE loss on training data and for unseen validation data after each epoch. There is no significant difference in convergence for either encoding but spiky outliers due to monte-carlo cross validation are visible.

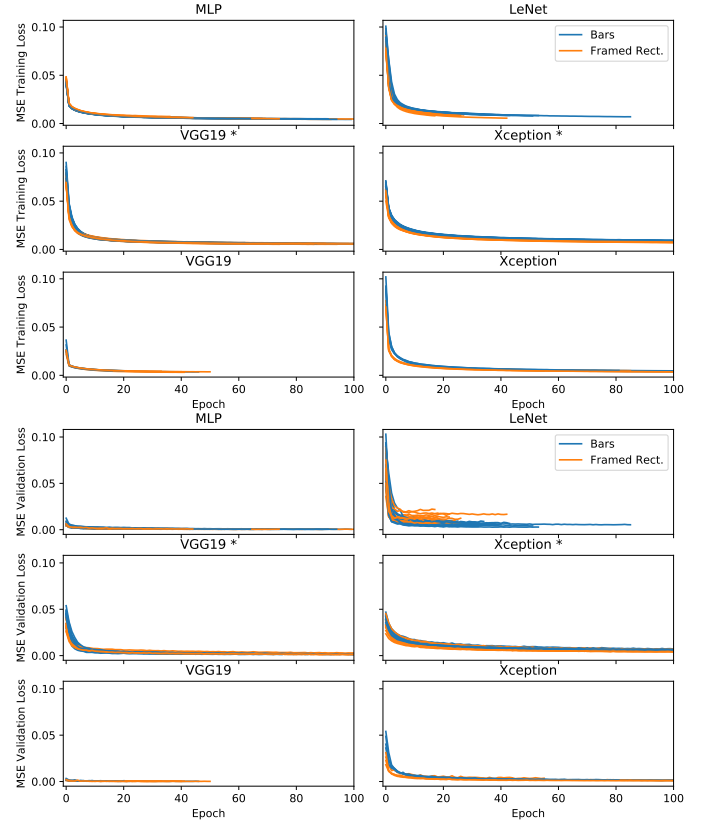
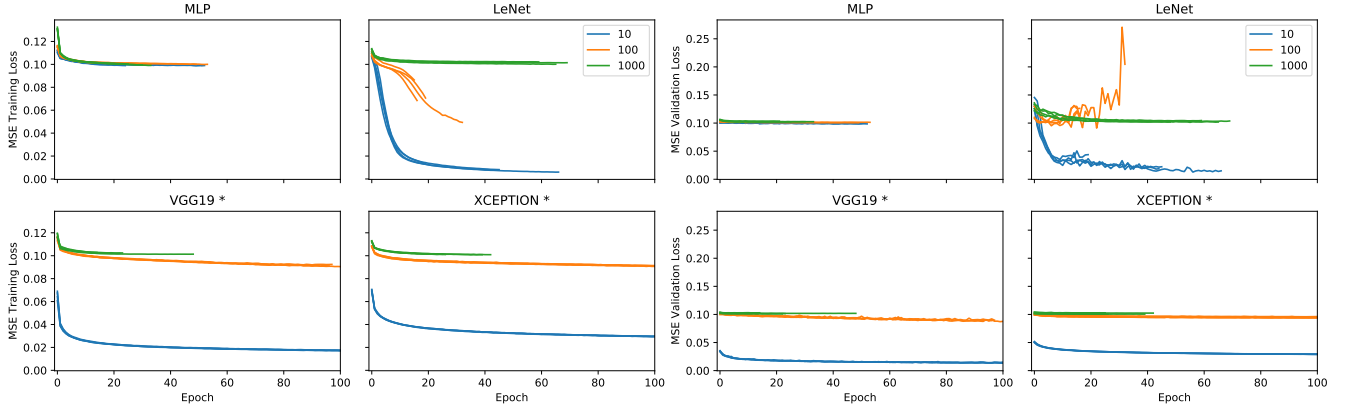
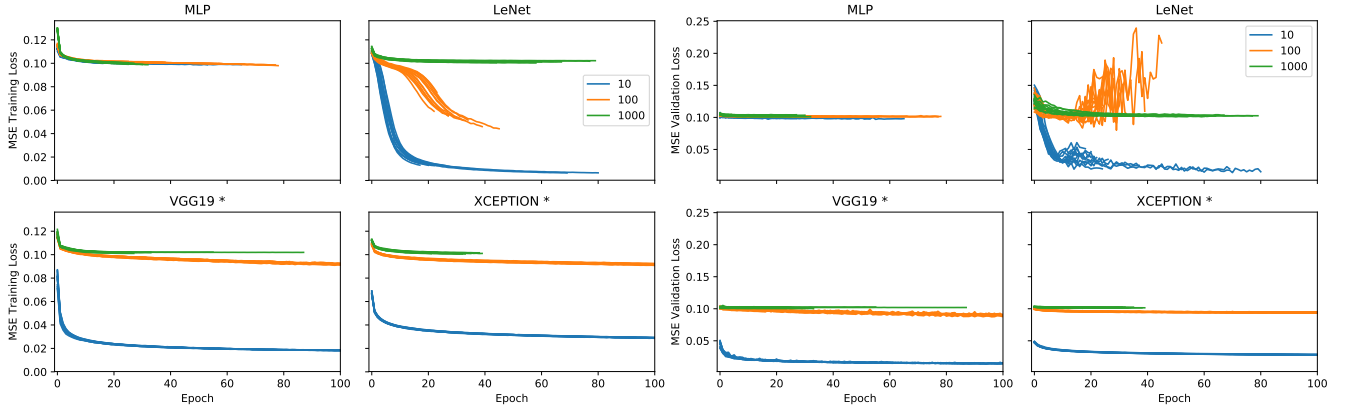


Fig. 12: **Loss plots for the bars-and-framed-rectangles experiment.** We visualize the MSE loss on training data and for unseen validation data after each epoch. There is no significant difference in convergence for either encoding.



(a) without noise



(b) with noise

Fig. 13: **Loss plots for the Weber-Fechner's law experiment.** We visualize the MSE loss on training data (left) and for unseen validation data (right) after each epoch (a) without noise and (b) with subtle 5% noise per pixel. There is no significant difference when noise is added. The LeNet network seems to overfit with Weber Base 100 in both cases even with dropout regularization.

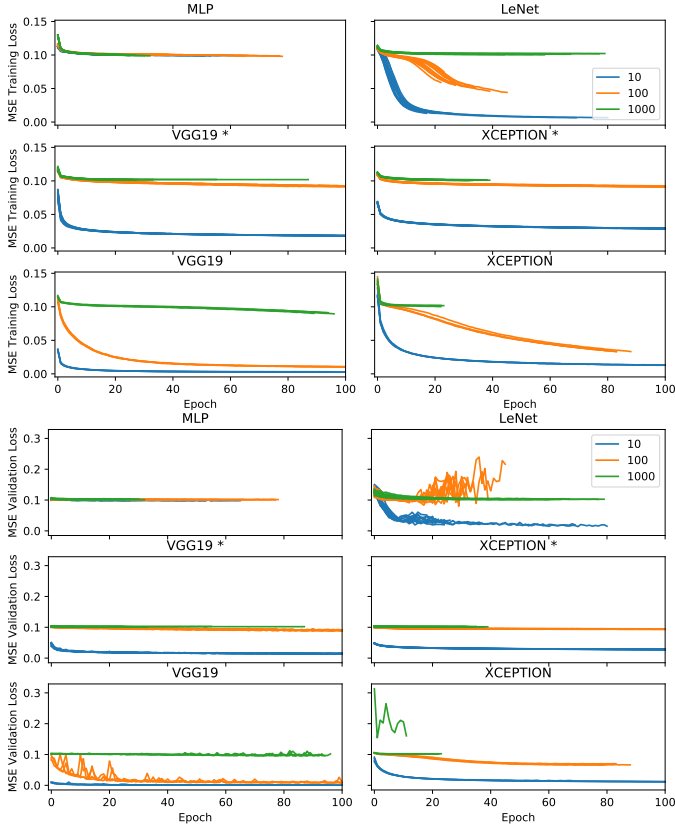
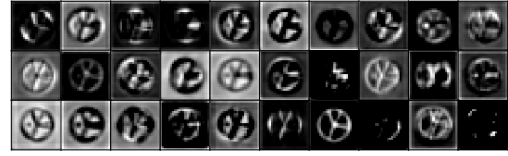
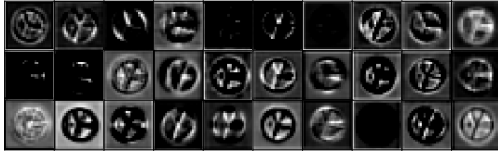
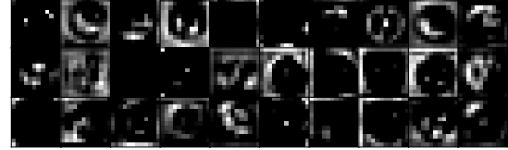
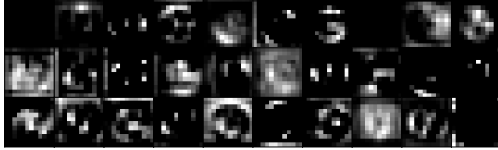


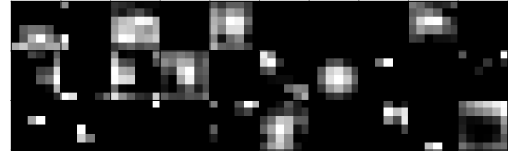
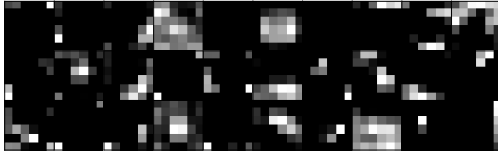
Fig. 14: **Loss plots for the Weber-Fechner’s law experiment including VGG19 and Xception.** We visualize the MSE loss on training data and for unseen validation data after each epoch. This plot includes the VGG19 and Xception networks trained from scratch.



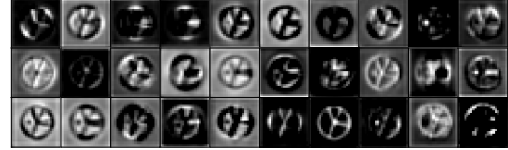
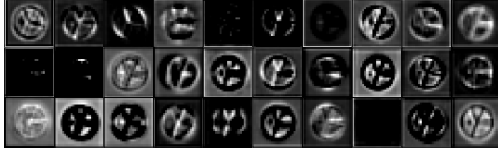
(a) VGG19 *, Block 3 (Conv. Layers 2+3)



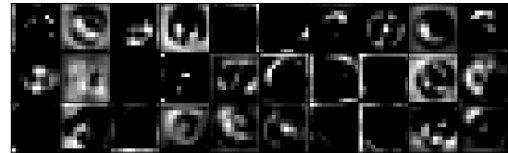
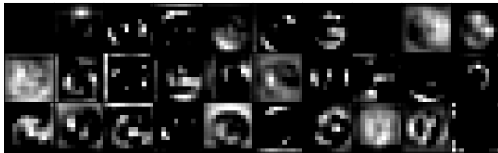
(b) VGG19 *, Block 4 (Conv. Layers 2+3)



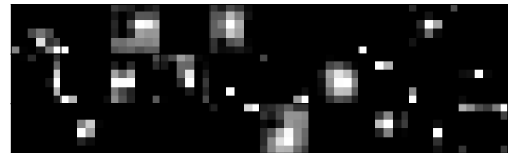
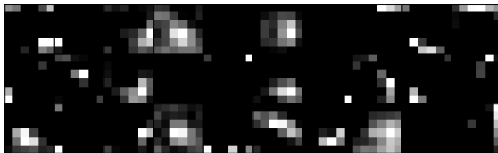
(c) VGG19 *, Block 5 (Conv. Layers 2+3)



(d) VGG19, Block 3 (Conv. Layers 2+3)

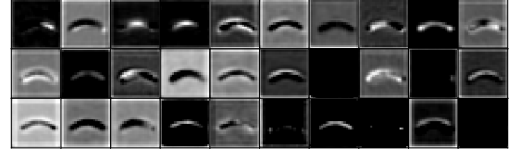
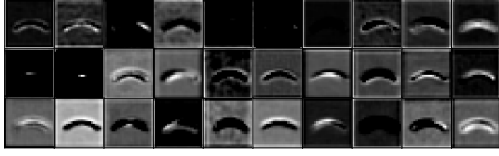


(e) VGG19, Block 4 (Conv. Layers 2+3)

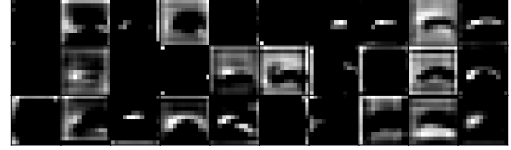
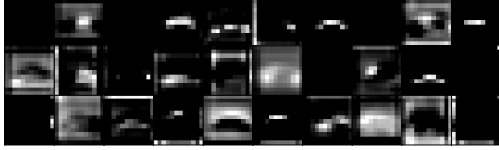


(f) VGG19, Block 5 (Conv. Layers 2+3)

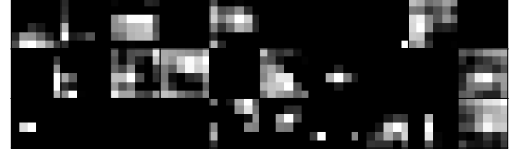
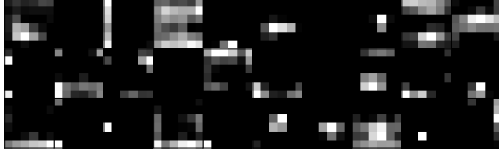
Fig. 15: **Convolutional Activation Maps for a Pie Chart.** (a)-(c) is VGG19 *, trained on ImageNet. The activation maps do not differ much which is surprising since VGG19 trained from scratch performs so much better in our experiments.



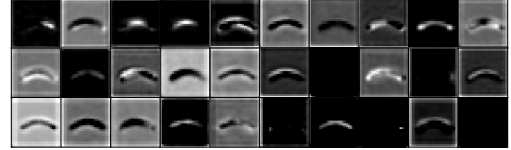
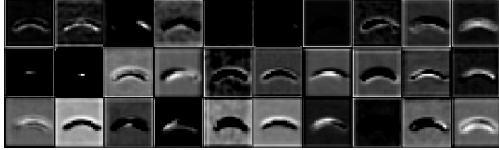
(a) VGG19 *, Block 3 (Conv. Layers 2+3)



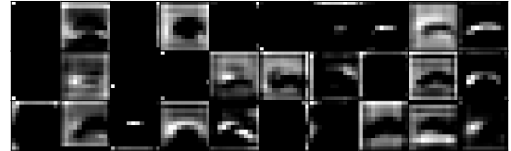
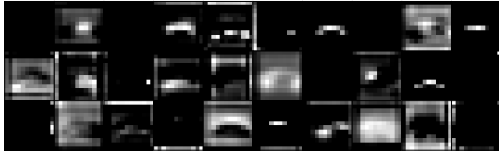
(b) VGG19 *, Block 4 (Conv. Layers 2+3)



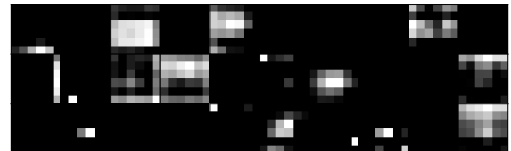
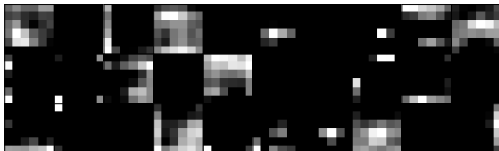
(c) VGG19 *, Block 5 (Conv. Layers 2+3)



(d) VGG19, Block 3 (Conv. Layers 2+3)



(e) VGG19, Block 4 (Conv. Layers 2+3)



(f) VGG19, Block 5 (Conv. Layers 2+3)

Fig. 16: **Convolutional Activation Maps for a Curvature stimuli.** (a)-(c) is VGG19 *, trained on ImageNet. The activation maps do not differ much which is surprising since VGG19 trained from scratch performs so much better in our experiments.

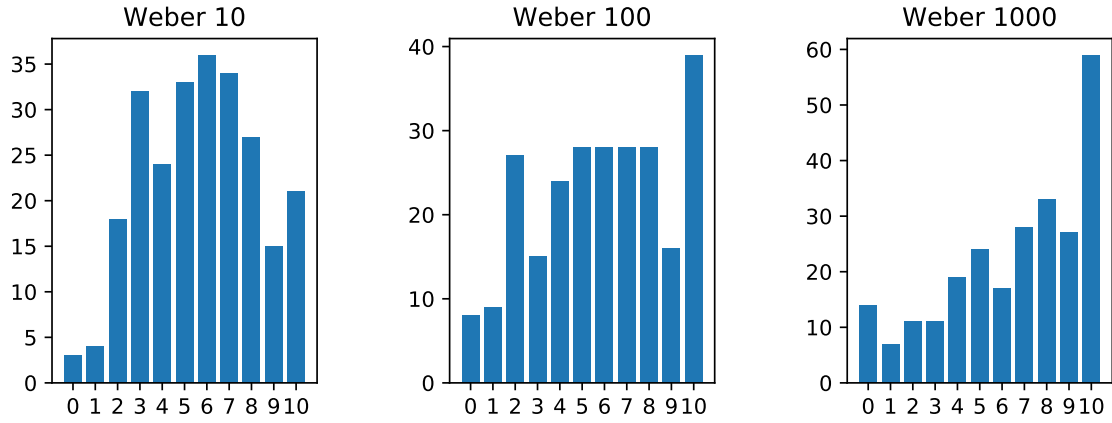


Fig. 17: **Weber's law point cloud experiment (E5): User responses.** The human results for 10, 100, 1,000 points exhibit skewed distributions. At 1,000 points, this task is virtually impossible, and so humans resort to simply guessing. However, they seem to 'guess high', leading to worse than random performance on the task.

Table 2: Mean squared error (MSE) for the elementary perceptual tasks experiment (E1), to accompany the midmean logistic absolute error metric (MLAE) measures in the main paper.

	MLP	LeNet	VGG19 *	VGG19	Xception *	Xception
Position Common Scale	0.03794 ± 0.01436	0.00106 ± 0.00028	0.00079 ± 0.00027	0.00014 ± 0.00009	0.00170 ± 0.00036	0.00074 ± 0.00034
Position Non-aligned Scale	0.02974 ± 0.00912	0.00097 ± 0.00043	0.00081 ± 0.00016	0.00021 ± 0.00013	0.00183 ± 0.00047	0.00060 ± 0.00006
Length	0.00267 ± 0.00053	0.01044 ± 0.00301	0.00046 ± 0.00050	0.00010 ± 0.00002	0.00142 ± 0.00021	0.00066 ± 0.00010
Direction	0.08359 ± 0.00645	0.01409 ± 0.00575	0.01014 ± 0.00212	0.00121 ± 0.00090	0.02342 ± 0.00556	0.00247 ± 0.00159
Angle	0.08092 ± 0.00953	0.01859 ± 0.00611	0.00415 ± 0.00087	0.00053 ± 0.00006	0.00607 ± 0.00064	0.00158 ± 0.00019
Area	0.00256 ± 0.00074	0.00383 ± 0.00171	0.00031 ± 0.00016	0.00010 ± 0.00004	0.00047 ± 0.00013	0.13955 ± 0.22880
Volume	0.00593 ± 0.00382	0.00405 ± 0.00567	0.00100 ± 0.00067	0.00084 ± 0.00065	0.00339 ± 0.00218	0.00346 ± 0.00050
Curvature	0.00343 ± 0.00076	0.00065 ± 0.00015	0.00024 ± 0.00003	0.00006 ± 0.00001	0.00047 ± 0.00010	0.00254 ± 0.00108
Shading	0.01861 ± 0.00811	0.00448 ± 0.00205	0.00078 ± 0.00067	0.00032 ± 0.00021	0.00303 ± 0.00138	0.00657 ± 0.00473

Table 3: Mean squared error (MSE) for the position-angle experiment (E2), to accompany the midmean logistic absolute error metric (MLAE) measures in the main paper

	MLP	LeNet	VGG19 *	VGG19	Xception *	Xception
Pie Chart	0.04725 ± 0.00177	0.02429 ± 0.00070	0.02492 ± 0.00143	0.00095 ± 0.00018	0.01371 ± 0.00078	0.00260 ± 0.00037
Bar Chart	0.00629 ± 0.00051	0.00322 ± 0.00029	0.00723 ± 0.00107	0.00076 ± 0.00012	0.00465 ± 0.00020	0.00109 ± 0.0001

Table 4: Mean squared error (MSE) for the position-length experiment (E3), to accompany the midmean logistic absolute error metric (MLAE) measures in the main paper.

	MLP	LeNet	VGG19 *	VGG19	Xception *	Xception
Type 1	0.03561 ± 0.01403	0.13262 ± 0.07927	0.04563 ± 0.02225	0.04259 ± 0.01621	0.07448 ± 0.04100	0.10515 ± 0.04370
Type 2	0.04270 ± 0.04115	0.07342 ± 0.04200	0.12256 ± 0.06932	0.03976 ± 0.02488	0.04624 ± 0.02606	0.08235 ± 0.02587
Type 3	0.06055 ± 0.02853	0.04998 ± 0.08381	0.09711 ± 0.04050	0.02746 ± 0.01733	0.03701 ± 0.01777	0.08443 ± 0.04828
Type 4	0.07275 ± 0.03090	0.05433 ± 0.04709	0.03922 ± 0.02143	0.03065 ± 0.02038	0.05543 ± 0.06636	0.06119 ± 0.06254
Type 5	0.06728 ± 0.02543	0.03583 ± 0.02154	0.12193 ± 0.05223	0.04314 ± 0.03622	0.04574 ± 0.01683	0.07167 ± 0.02072
Multi	0.02624 ± 0.01729	0.04419 ± 0.02724	0.07631 ± 0.02532	0.01756 ± 0.00931	0.03850 ± 0.02710	0.08365 ± 0.02061

Table 5: Mean squared error (MSE) for the bars and framed rectangles experiment (E4), to accompany the midmean logistic absolute error metric (MLAE) measures in the main paper

	MLP	LeNet	VGG19 *	VGG19	Xception *	Xception
Framed Rectangles	0.00065 ± 0.00012	0.01343 ± 0.00484	0.00156 ± 0.00029	0.00032 ± 0.00011	0.00437 ± 0.00069	0.00155 ± 0.00062
Bars	0.00060 ± 0.00012	0.00553 ± 0.00196	0.00173 ± 0.00037	0.00043 ± 0.00024	0.00531 ± 0.00073	0.00145 ± 0.00053

Table 6: Mean squared error (MSE) for the Weber's law point cloud experiment (E5), to accompany the midmean logistic absolute error metric (MLAE) measures in the main paper.

	MLP	LeNet	VGG19 *	VGG19	Xception *	Xception
10 points	0.10126 ± 0.00062	0.13759 ± 0.03183	0.09031 ± 0.00130	0.00899 ± 0.00150	0.09354 ± 0.00078	0.06830 ± 0.00207
100 points	0.10250 ± 0.00057	0.10259 ± 0.00065	0.10202 ± 0.00038	0.09835 ± 0.00267	0.10174 ± 0.00034	0.10818 ± 0.01003
1,000 points	0.09946 ± 0.00095	0.03436 ± 0.01137	0.01477 ± 0.00059	0.00076 ± 0.00008	0.02610 ± 0.00026	0.01041 ± 0.00258