Appendix A: Experimental details

Image classifiers. For each image classifier in Section 5.1 we used the same network structure: Two layers of convolution-ReLU-max-pooling, each with kernels of size 5, and no padding, followed by one fully connected layer, and a final linear layer that outputs the unnormalized softmax probabilities. For MNIST we used 32 filters in each convolutional layer, and for SVHN and CIFAR we used 64 and 128 filters for CONV1 and CONV2 respectively. The fully connected layer takes the vectorized feature maps of CONV2 and maps to FC3 which has 256 units for all datasets. We used dropout at every layer, and Adam optimizer with learning rate 3×10^{-4}. We trained the networks for a maximum of 250000 steps, and used early stopping with respect to the validation accuracy.

For the AffNIST experiments in Section 5.2 we use three image classifiers, one with global spatial pooling, one without global spatial pooling, and a network that replaces convolutional layers with fully connected layers. The network without global spatial pooling is identical to the MNIST network used in Section 5.1. The network with global spatial pooling is identical except that it applies global max pooling to the CONV2 feature maps. The output of this operation is a vector with the same number of dimensions as there are channels in the CONV2 feature maps. This vector is then connected using a fully-connected layer to the 256 unit FC3 features. The fully connected network replaces CONV1 and CONV2 with fully connected layers each with 2048 units. The training details are the same as for the other image classifiers.

PixelCNN++. For all inversion networks we used the PixelCNN++ architecture detailed in (Salimans et al., 2017). The architecture consists of six blocks of residual layers, with spatial downsampling using strided convolutions between the first, second and third blocks, and spatial upsampling using strided transpose convolutions between the fourth, fifth and sixth blocks. In order to preserve high resolution information skip connections are employed between corresponding downsampling and upsampling blocks. In order to reduce the cost of training the models, we used three residual layers in each block rather than the five specified in the original architecture. We use 64 filters in the convolutional layers for MNIST and 196 for SVHN and CIFAR. For SVHN and CIFAR we used the discretized mixture of logistics described in (Salimans et al., 2017) with 10 mixture components for the conditional pixel likelihood. For MNIST we used a 256-way softmax distribution over the discrete pixel values as in the original PixelCNN (van den Oord et al., 2016c) as we found it to be much more effective in practice.

We used weight normalization with data-dependent initialization (Salimans and Kingma, 2016), and trained our models using Adam optimizer with initial learning rate 10^{-3} and a learning rate decay of 0.9999 for a maximum of 250000 weight updates. Again we used early stopping, but found that in general the validation performance continued to improve for the duration of training. To condition on vector representations FC3 and LOGITS we linearly projected the context vector to biases that are added to feature maps in each residual layer. For spatial representations CONV1 and CONV2 we resize the context to match the PixelCNN++ feature maps and use 1×1 convolutions to project to spatially-structured biases that are added to the feature maps. We used a single dropout layer in each PixelCNN++ residual block for all networks. For CONV1 inversion models on SVHN and CIFAR we used dropout rate 0.1, and for all other networks we used dropout rate 0.5. We applied an additional dropout layer with dropout rate 0.2 to the outputs of the linear projection of the context for the CIFAR-FC3 inversion model.

Appendix B: More details on mutual information estimation in neural networks

Here we describe some alternative approaches to mutual information estimation in neural networks not covered in Section 3.3.

Discretization. Schwartz-Ziv and Tishby (2017) discretize tanh activations in a fully-connected neural network each into 30 equally sized bins to form a discrete empirical distribution $p(\mathbf{h}|\mathbf{x})$. In experiments with a known distribution of discrete inputs $p(\mathbf{x})$, they exactly compute the mutual information between the inputs and the discretized layer activations by averaging over settings of $\mathbf{x}$ and $\mathbf{h}$.

$$I(\mathbf{x};\mathbf{h}) = \sum\nolimits_{\mathbf{x},\mathbf{h}} p(\mathbf{x},\mathbf{h}) \log\left(\frac{p(\mathbf{x},\mathbf{h})}{p(\mathbf{x})p(\mathbf{h})}\right) \tag{12}$$

Saxe et al., (2018) note that the networks of interest do not operate on the discretized values, and that the binning is used solely for mutual information calculations. Moreover, there are many possible ways of binning potentially unbounded activations such as ReLUs, and the choice can significantly impact the mutual information estimates.

Non-parametric entropy estimation. Saxe et al., (2018) similarly obtain an approximate bound on the mutual information by estimating the entropy of activations with additive noise $H(\mathbf{h}^{\epsilon})$. They use the

estimator of Kraskov et al. (2004) that makes use of distances between nearest neighbours in a collection of samples. The entropy estimator is:

$$\hat{H}(\mathbf{h}) = \frac{D}{N}\sum\nolimits_i \log(r_i + \epsilon) + \frac{D}{2}\log(\pi) \tag{13}$$

$$- \log\Gamma(D/2+1) + \psi(N) - \psi(k), \tag{14}$$

where D is the dimensionality of $\mathbf{h}$, N is the number of samples, r_i is the distance between sample i and its k'th nearest neighbour, Γ is the Gamma function, and ψ is the digamma function. As with the KDE-based approach described in Section 3.3, this non-parametric estimate may be problematic for analysis of network layers with very many units.