# TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2018

## Entropy and Compressibility

## Rate-Distortion Autoencoders

# Big Picture: Cross-Entropy as a Data Rate

For a distribution $P(y)$ on a discrete set $\mathcal{Y}$, the entropy $H(P)$, when measured using $\log_2$ rather than $\ln$, gives the number of bits needed on average, when drawing from $P$, to represent the elements of $\mathcal{Y}$.

The cross-entropy $H(P, Q)$ give the number of bits used to code for items drawn from $P$ but using the code defined by $Q$.

Cross-entropy gives the "data rate" when transmitting codes for items drawn from $P$ but using the code defined by $Q$.

# Entropy and Compressibility

Let $S$ be a finite set.

Let $z$ be a compression (or coding) function assigning a bit string $z(y)$ to each $y \in S$.

The compression function $z$ is called *prefix-free* if for $y' \neq y$ we have that $z(y')$ is not a prefix of $z(y)$.

# Prefix-Free Codes as Probabilities

A prefix-free code defines a binary branching tree — branch on the first code bit, then the second, and so on.

For a prefix-free code, only the leaves of this tree can be labeled with the elements of $S$.

The code defines a probability distribution on $S$ by randomly selecting branches.

We have $P_z(y) = 2^{-|z(y)|}$.

# Bits vs. Nats

We have that $|z(y)|$ is a number of bits.

We can define entropy in units of bits by

$$H_2(y) = E_y \ -\log_2 P(y) = H(y)/(\ln \ 2)$$

If $y$ is uniformly distributed over 8 values then $H_2(y)$ is 3 bits.

We have that $H_2(y)$ is a number of bits while $H(y)$ is a number of "nats".

# The Source Coding (compression) Theorem

(1) There exists a prefix-free code $z$ such that
$$|z(y)| <= (-\log_2 \text{Pop}(y)) + 1$$
and hence
$$E_{y \sim \text{Pop}} |z(y)| \leq H_2(\text{Pop}) + 1$$

(2) For any prefix-free code $z$

$$E_{y \sim \text{Pop}} |z(y)| \geq H_2(\text{Pop})$$

# Code Construction

We construct a code by iterating over $y \in S$ in order of decreasing probability (most likely first).

For each $y$ select a code word $z(y)$ (a tree leaf) with length (depth)

$$|z(y)| = \lceil -\log_2 \operatorname{Pop}(y) \rceil$$

and where $z(y)$ is not an extension of (under) any previously selected code word.

# Code Existence Proof

At any point before coding all elements of $S$ we have

$$\sum_{y \in \text{Defined}} 2^{-|z(y)|} \leq \sum_{y \in \text{Defined}} \text{Pop}(y) < 1$$

Therefore there exists an infinite descent into the tree that misses all previous code words.

Hence there exists a code word $z(x)$ not under any previous code word with $|z(x)| = \lceil -\log_2 \text{Pop}(y) \rceil$.

Furthermore $z(x)$ is at least as long as all previous code words and hence $z(x)$ is not a prefix of any previously selected code word.

# No Better Code Exists

Let $z$ be an arbirtary coding.

$$E_y \ |z(y)| \ = \ E_y \ -\log_2 P_z(y)$$

$$= \ H_2(\mathrm{Pop}, P_z)$$

$$= \ H_2(\mathrm{Pop}) + KL_2(\mathrm{Pop}, P_Z)$$

$$\geq \ H_2(\mathrm{Pop})$$

# Big Picture: Differential Entropy is Always Infinite

For a continuous set $\mathcal{Y}$ (such as the unit interval on real numbers) and for any probability density (smooth measure) on $\mathcal{Y}$ the probability of any single point $y \in \mathcal{Y}$ is zero.

This implies that the entropy is infinite — it takes an infinite number of bits to represent a random real number.

Cross-entropy between continuous densities is also always infinite and <span style="color:red">differential cross-entropy loss is not meaningful.</span>

# Differential Entropy

Consider a continuous density $p(x)$. For example

$$p(x) = \frac{1}{\sqrt{2\pi}\ \sigma}\ e^{\frac{-x^2}{2\sigma^2}}$$

Differential entropy is often defined as

$$H(p) \doteq \int \left( \ln \frac{1}{p(x)} \right) p(x) dx$$

# Finite Differential Entropy is Not Meaningful

$$H(\mathcal{N}(0, \sigma)) = + \int \left( \ln(\sqrt{2\pi}\sigma) + \frac{x^2}{2\sigma^2} \right) p(x)dx$$

$$= \ln(\sigma) + \ln(\sqrt{2\pi}) + \frac{1}{2}$$

But if we take $y \doteq x/2$ we get $H(y) = H(x) - \ln 2$.

Also for $\sigma << 1$, we get $H(p) < 0$

Hence differential entropy then depends on the choice of units — a distributions on lengths will have a different entropy when measuring in inches than when measuring in feet.

# Differential Entropy is Always Infinite

Consider quantizing the the real numbers into bins.

A continuous probability densisty $p$ assigns a probability $p(B)$ to each bin.

As the bin size decreases toward zero the entropy of the bin distribution increases toward $\infty$.

A meaningful convention is that $H(p) = +\infty$ for any continuous density $p$.

# Differential KL-divergence is Meaningful

$$KL(p, q) = \int \left( \ln \frac{p(x)}{q(x)} \right) p(x) dx$$

This integral can be computed by dividing the real numbers into bins and computing the $KL$ divergence between the distributions on bins.

The KL divergence between the bin distribution typically approaches a finite limit as the bin size goes to zero.

# KL-Divergence can also be Infinite

$$KL(p,q) = E_{x \sim p} \ \log \frac{p(x)}{q(x)}$$

In either the discrete or continuous case, if a set is assigned nonzero probability by $p$ but zero probability by $q$ then $KL(p,q) = +\infty$.

If every set assigned nonzero probability by $p$ is also assigned nonzero probability by $q$ then we say that $p$ is absolutely continuous with respect to $q$.

# Big Picture: Unsupervised Learning



■ "Pure" Reinforcement Learning (cherry)
  ▶ The machine predicts a scalar reward given once in a while.
  ▶ **A few bits for some samples**

■ Supervised Learning (icing)
  ▶ The machine predicts a category or a few numbers for each input
  ▶ Predicting human-supplied data
  ▶ **10→10,000 bits per sample**

■ Unsupervised/Predictive Learning (cake)
  ▶ The machine predicts any part of its input for any observed part.
  ▶ Predicts future frames in videos
  ▶ **Millions of bits per sample**

■ (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

Consider the power of BERT.

We would like to density estimation for images and sound waves.

# The Big Picture: Rate-Distortion Loss

For density estimation of naturally occurring continuous distributions we consider a parameterized "rounding" operation mapping $y$ to $\tilde{y}_\Phi(y)$ with $\tilde{y} \in \mathcal{Y}$ for $\mathcal{Y}$ discrete.

We then define rate-distortion loss

$$\mathcal{L}(\Phi) = E_{y \sim \text{Pop}} \left( -\ln P_\Phi(\tilde{y}_\Phi(y)) \right) + \lambda D(y, \tilde{y}_\Phi(y))$$

where $D(y, \tilde{y})$ is some "distortion function" measuring a distance between $y$ and $\tilde{y}$.

# The Rate-Distortion Tradeoff

$$\mathcal{L}(\Phi) = E_{y\sim\mathrm{Pop}}\left(-\ln P_\Phi(\tilde{y}_\Phi(y))\right) + \lambda D(y, \tilde{y}_\Phi(y))$$

The first term is just cross-entropy loss which we are calling the "rate". This terminology is explained below.

The meta-parameter $\lambda$ controls the trade off between rate and distortion.

# Common Distortion Functions

$$\Phi^* = E_{y \sim \text{Pop}} \left( -\ln P_\Phi(\tilde{y}_\Phi(y)) \right) + \lambda D(y, \tilde{y}_\Phi(y))$$

It is common to take

$$D(y, \tilde{y}) = ||y - \tilde{y}||^2 \qquad (L_2)$$

or

$$D(y, \tilde{y}) = ||y - \tilde{y}||_1 \qquad (L_1)$$

# Rate-Distortion Autoencoders

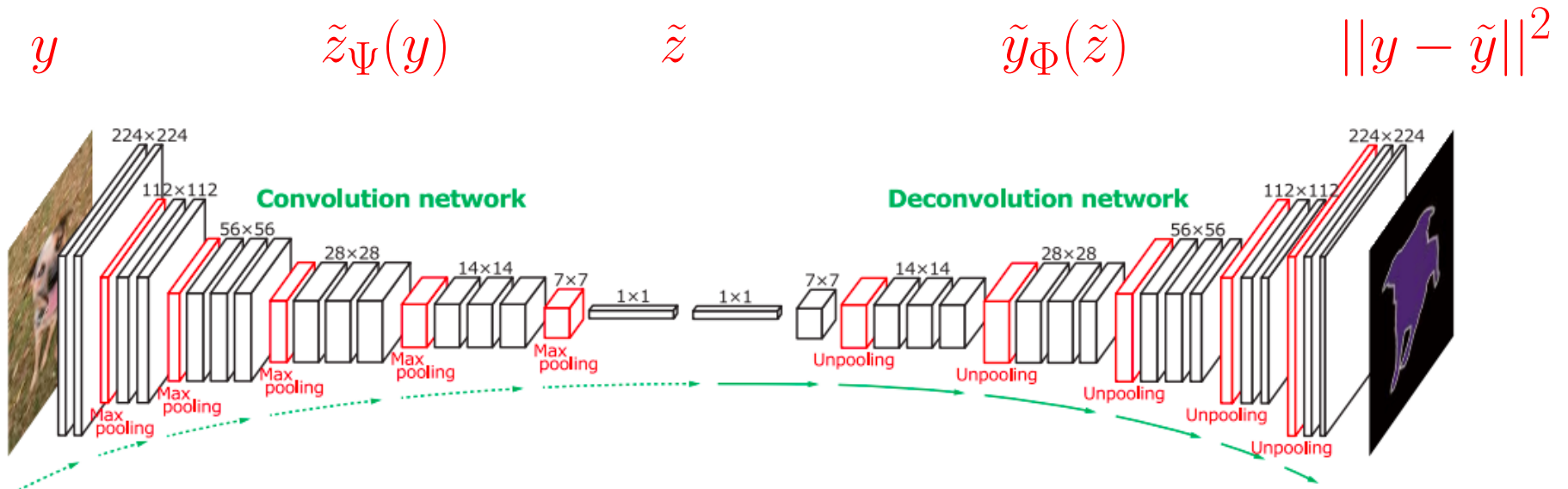Given a continuous signal $y$ we can compress it into a (discrete) bit string $\tilde{z}_\Phi(y)$.

We let $\tilde{y}_\Phi(\tilde{z}_\Phi(y))$ be the decompression of $\tilde{z}_\Phi(y)$.

Rate-Disrtion loss can then be written as

$$\mathcal{L}(\Phi) = E_{y \sim \mathrm{Pop}} \, |\tilde{z}_\Phi(y)| + \lambda D(y, \tilde{y}(\tilde{z}(y)))$$

# A Case Study in Image Compression

**End-to-End Optimized Image Compression, Balle, Laparra, Simoncelli, ICLR 2017.**

$$y \qquad \tilde{z}_\Psi(y) \qquad \tilde{z} \qquad \tilde{y}_\Phi(\tilde{z}) \qquad ||y - \tilde{y}||^2$$



The model described here has been simplified from the original.

# JPEG at 4283 bytes or .121 bits per pixel



JPEG, 4283 bytes (0.121 bit/px), PSNR: 24.85 dB/29.23 dB, MS-SSIM: 0.8079

# JPEG 2000 at 4004 bytes or .113 bits per pixel



JPEG 2000, 4004 bytes (0.113 bit/px), PSNR: 26.61 dB/33.88 dB, MS-SSIM: 0.8860

# Deep Autoencoder at 3986 bytes or .113 bits per pixel



**Proposed method**, 3986 bytes (0.113 bit/px), PSNR: 27.01 dB/34.16 dB, MS-SSIM: 0.9039

# The Encoder $z_\Phi(y)$

This paper uses a three layer CNN for the encoder.

The first layer is computed stride 4.

The two remaining layers are computed stride 2.

# The number of numbers
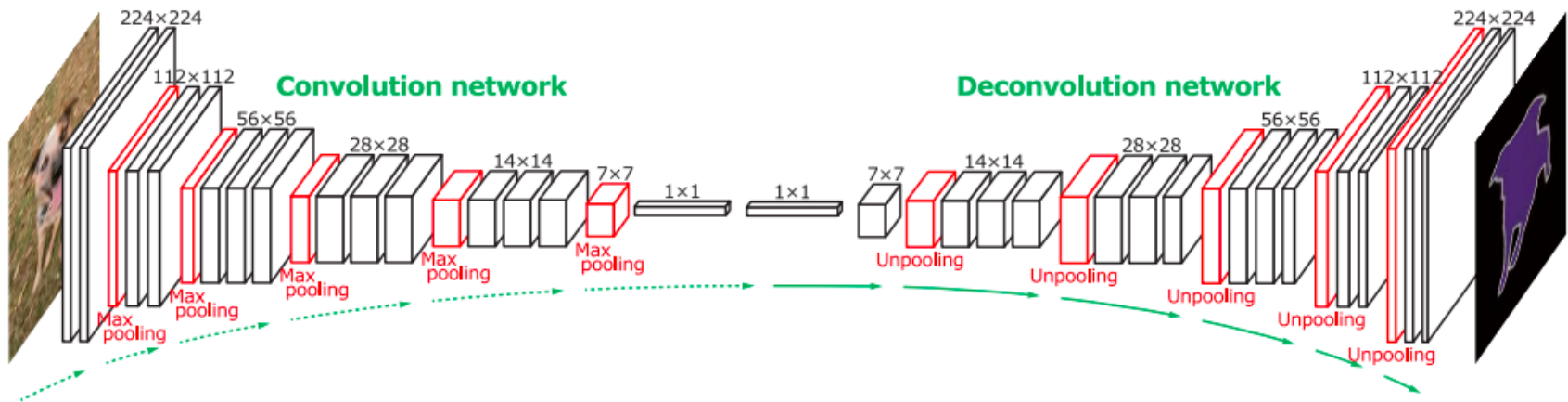
The first layer is computed stride 4.

The next two layers are computed stride 2.

Final image dimension is reduced by a factor of 16 with 192 channels per pixel (192 channels is for color images).

$$192 < 16 \times 16 \times 3 = 768$$

The final values $z[x, y, i]$ are rounded to integers $\tilde{z}[x, y, i]$.

# Increasing Spatial Dimension in Decoding



[Hyeonwoo Noh et al.]

In the ICLR 17 paper the deconvolution network has the shape as the input CNN but with independent parameters.

# Increasing Spatial Dimensions in Deconvolution

Consider a stride 2 convolution

$$L_{\ell+1}[x, y, j] = \sigma \left( \sum_{\Delta x, \Delta y, i} W[\Delta x, \Delta y, i, j] L_\ell[2x + \Delta x, 2y + \Delta y, i] \right)$$

For deconvolution we use stride 1 with 4 times the features.

$$L'_\ell[x, y, i] = \sigma \left( \sum_{\Delta x, \Delta y, j} W[\Delta x, \Delta y, j, i] L'_{\ell+1}[x + \Delta x, y + \Delta y, j] \right)$$

The channels at each $L'_\ell[x, y]$ are divided among four higher resolution pixels.

This is done by a simple reshaping of $L'_\ell[x, y, i]$.

# Rate-Distortion Autoencoders

$$\Phi^* = \operatorname*{argmin}_{\Phi} \; E_{y \sim \mathrm{Pop}} \; |\tilde{z}_\Phi(y)| + \lambda D(y, \tilde{y}_\Phi(z_\Phi(y)))$$

Oops: Because of rounding, $\tilde{z}_\Phi(y)$ is discrete and the gradients are zero.

Note, however that the rate-distortion loss is measurable.

We will approximate the gradient descent but still be able to measure loss.

# Rate-Distortion Autoencoders

$$\Phi^* \quad = \underset{\Phi}{\arg\min} \; \mathcal{L}_{\text{rate}}(\Phi) + \lambda \mathcal{L}_{\text{dist}}(\Phi)$$

$$\mathcal{L}_{\text{rate}}(\Phi) = E_{y \sim \text{Pop}} \; |\tilde{z}_\Phi(y)|$$

$$\mathcal{L}_{\text{dist}}(\Phi) = E_{y \sim \text{Pop}} \; D(y, \tilde{y}_\Phi(\tilde{z}_\Phi(y)))$$

We will consider differentiable approximations to both $\mathcal{L}_{\text{rate}}$ and $\mathcal{L}_{\text{dist}}$.

# A Differentiable Approximation of $\mathcal{L}_{\text{rate}}$

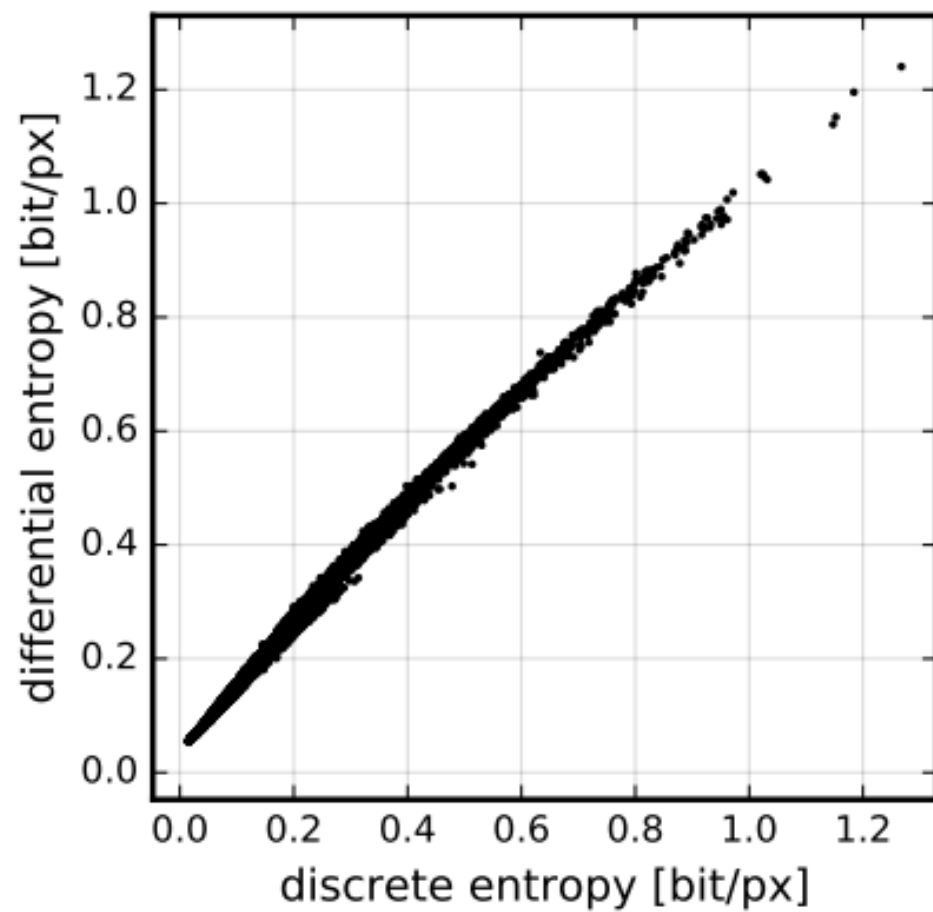$$\mathcal{L}_{\text{rate}}(\Phi) = E_{y \sim \text{Pop}} |\tilde{z}_\Phi(y)|$$

Recall that $\tilde{z}(y)$ is a rounding of a continuous tensor $z[x, y, i]$.

We can use the differentiable approximation

$$|\tilde{z}_\Phi(y)| \approx \sum_{x,y,i} \max(0, \log_2 z_\Phi[x, y, i])$$

This can be viewed as approximating a discrete entropy with differential entropy.

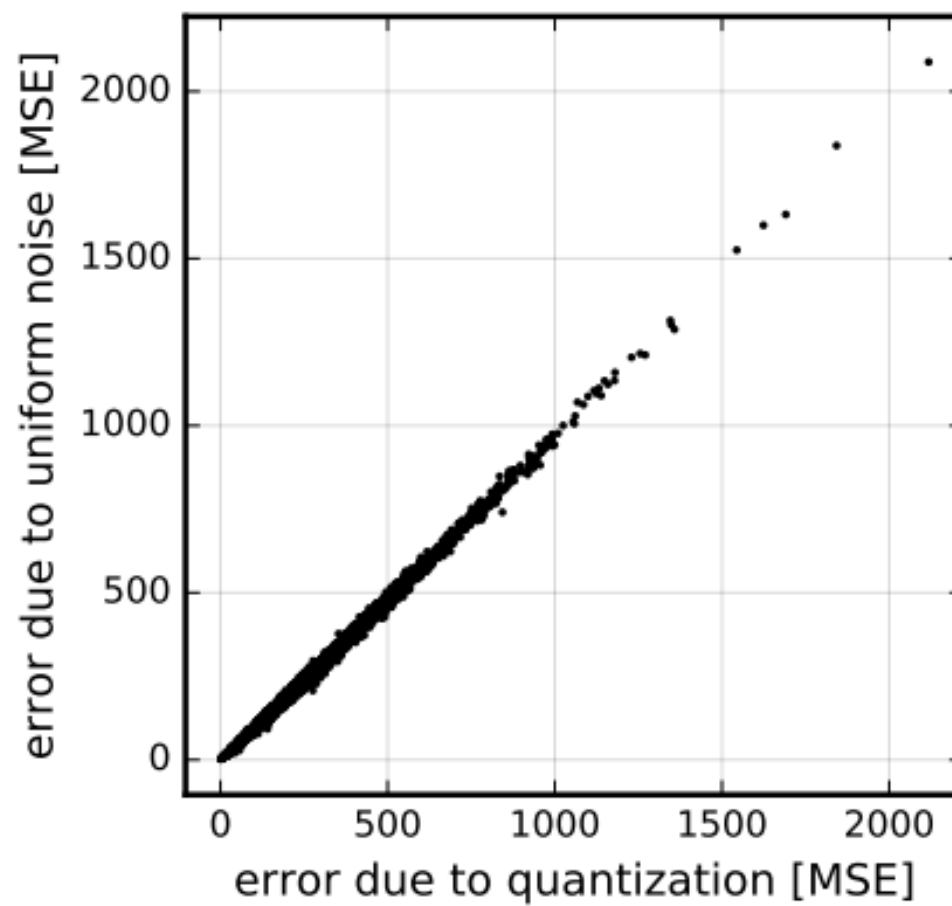# Differential Entropy vs. Discrete Entropy

# A Differentiable Approximation of $\mathcal{L}_{\mathrm{dist}}$

$$\mathcal{L}_{\mathrm{dist}}(\Phi) \;=\; E_{y \sim \mathrm{Pop}}\; D(y, \tilde{y}_\Phi(\tilde{z}_\Phi(y)))$$

$$\approx\; E_{y,\epsilon}\; D(y,\; \tilde{y}(z_\Phi(y) + \epsilon))$$

Here $\epsilon$ is a noise tensor with $\epsilon[x, y, i]$ drawn uniformly form $(-1/2, 1/2)$.

# Noise vs. Rounding

# Varying the Level Of Compression

$$\Phi^* = \operatorname*{argmin}_{\Phi} \ E_{y \sim \text{Pop}} |\tilde{z}_\Phi(y)| + \frac{1}{2}\lambda ||y - \tilde{y}_\Phi(\tilde{z}_\Phi(y))||^2$$

Different levels of compression correspond to different values of $\lambda$.

In all levels of compression we replace 768 numbers by 192 numbers.

Higher levels of compression result in smaller integer values in the 192 numbers.

# Conditional Rate-Distortion Autoencoders

$$\Phi^* = \operatorname*{argmin}_{\Phi} \; E_{(x,y)\sim\mathrm{Pop}} \; |\tilde{z}_\Phi(y|x)| + \lambda D(y \mid \tilde{y}_\Phi(x, \tilde{z}_\Phi(y|x)))$$

# Colorization



$$\Phi^* = \operatorname*{argmin}_{\Phi} E_{(x,y)\sim\mathrm{Pop}} \; |\tilde{z}_\Phi(y|x)| + \frac{1}{2}\lambda ||y - \tilde{y}_\Phi(x, \tilde{z}_\Phi(y|x))||^2$$

If the image can be segmented based on $x$ then $\tilde{z}_\Phi(y|x)$ can be a specification of color of each segment — this would be very compact.

# END