TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2018

Rate-Distortion Autoencoders

Rate-Distortion Autoencoders

Given an image or sound wave y we can compress it into a coded form $C_\Phi(y)$.

Let $|C_\Phi(y)|$ denote the number of bits in the compressed form $C_\Phi(y)$.

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

We let $D(y, \hat{y})$ be a measure of the difference between y and $\hat{y}$ (the loss or "distortion").

Rate-Distortion Autoencoders

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

It is common to take

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

or

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

Conditional Rate-Distortion Autoencoders

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \quad E_{(x,y)\sim \mathrm{Pop}} \quad |C_\Phi(y|x)| + \lambda D(y \mid \hat{y}_\Phi(x, C_\Phi(y|x)))$$

Colorization

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

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

A Case Study in Image Compression

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

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}}\, E_{y\sim \mathrm{Pop}}\, |C_\Phi(y)| + \frac{1}{2}\lambda||y - \hat{y}_\Phi(C_\Phi(y))||^2$$

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

Proposed Method 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 $C_\Phi(y)$

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.

They use a generilized divisive normalization (GDN) layer rather than an activation function.

$$y'[i,j,c] = \frac{y[i,j,c]}{\left(\beta_c + \sum_{c'} \gamma_{c,c'}\, y[i,j,c']^2\right)^{1/2}}$$

β_c and $\gamma_{c,c'} = \gamma_{c',c}$ are trained.

The number of numbers

The first layer is computed stride 4.

The next two layers are computed stride 2.

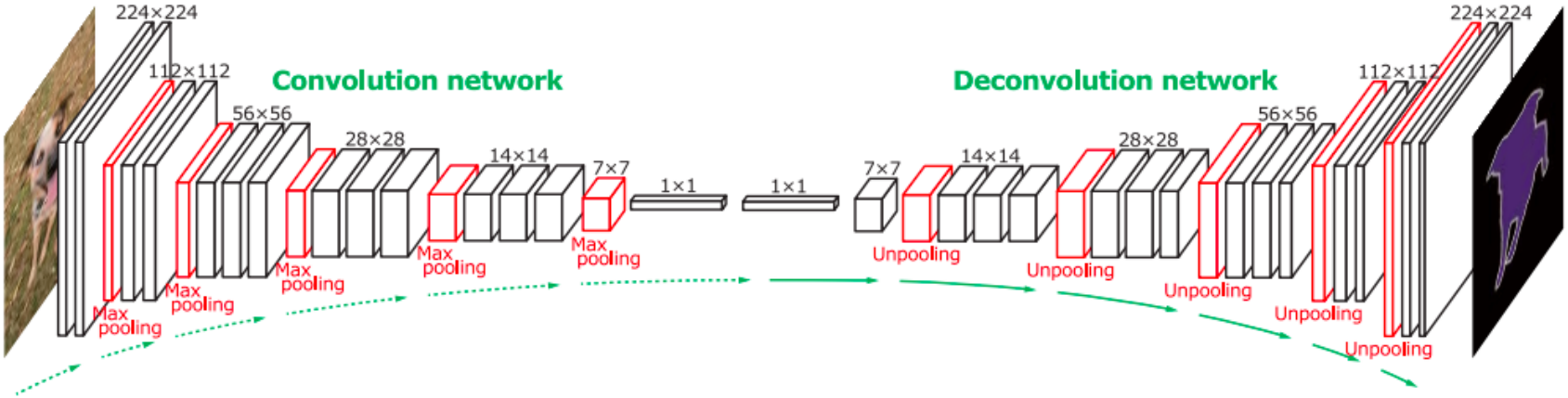
Final image dimension is reduced by a factor of 16 with 192 channels per pixel (192 channels is for color images).

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

These 192 numbers are rounded to integers.

The 192 integers are coded losslessly using P_{Θ}^{code}.

Increasing Spatial Dimension in Decoding



[Hyeonwoo Noh et al.]

Increasing Spatial Dimensions in Deconvolution

Consider a stride 2 convolution

$$\begin{aligned} y[i,j,c_y] &= W[\Delta i, \Delta j, c_x, c_y] x[2i + \Delta i, 2j + \Delta j, c_x] \\ y[i,j,c_y] &\texttt{ += } B[c_y] \end{aligned}$$

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

$$\begin{aligned} \hat{x}[i,j,c_{\hat{x}}] &= W'[\Delta i, \Delta j, c_{\hat{y}}, c_{\hat{x}}] \hat{y}[i + \Delta i, j + \Delta j, c_{\hat{x}}] \\ \hat{x}[i,j,c_{\hat{x}}] &\texttt{ += } B[c_{\hat{x}}] \end{aligned}$$

The channels at each lower resolution pixel $\hat{x}[i,j]$ are divided among four higher resolution pixels.

This is done by a simple reshaping of $\hat{x}$.

The Decoder

This is a deconvolution network of the same architecture with independent parameters.

There is a special parameterization of the "inverter" for the normalization layer.

Rounding the Numbers

We let $C_\Phi(x)$ be the unrounded numerical representation and $\tilde{C}_\Phi(x)$ be the result of rounding.

$$\tilde{C}_\Phi(x)_i = \text{round}(C_\Phi(x)_i) = \lfloor C_\Phi(x)_i + 1/2 \rfloor$$

Each integer channel of the final layer is coded independently.

Context-based adaptive binary arithmetic coding framework (CABAC; Marpe, Schwarz, and Wiegand, 2003).

Training

We now have the optimization problem

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \; E_{y \sim \mathrm{Pop}} - \log_2 P_\Phi(\tilde{C}_\Phi(y)) + \lambda ||y - \hat{y}_\Phi(\tilde{C}_\Phi(y))||^2$$

Issue: The rounding causes the gradients for Φ to be zero.

Modeling Rounding with Noise

$$\Phi^* = \underset{\Phi}{\text{argmin}}\ E_{y\sim \text{Pop}} - \log_2 P_\Phi(\tilde{C}_\Phi(y)) + \lambda||y - \hat{y}_\Phi(\tilde{C}_\Phi(y))||^2$$

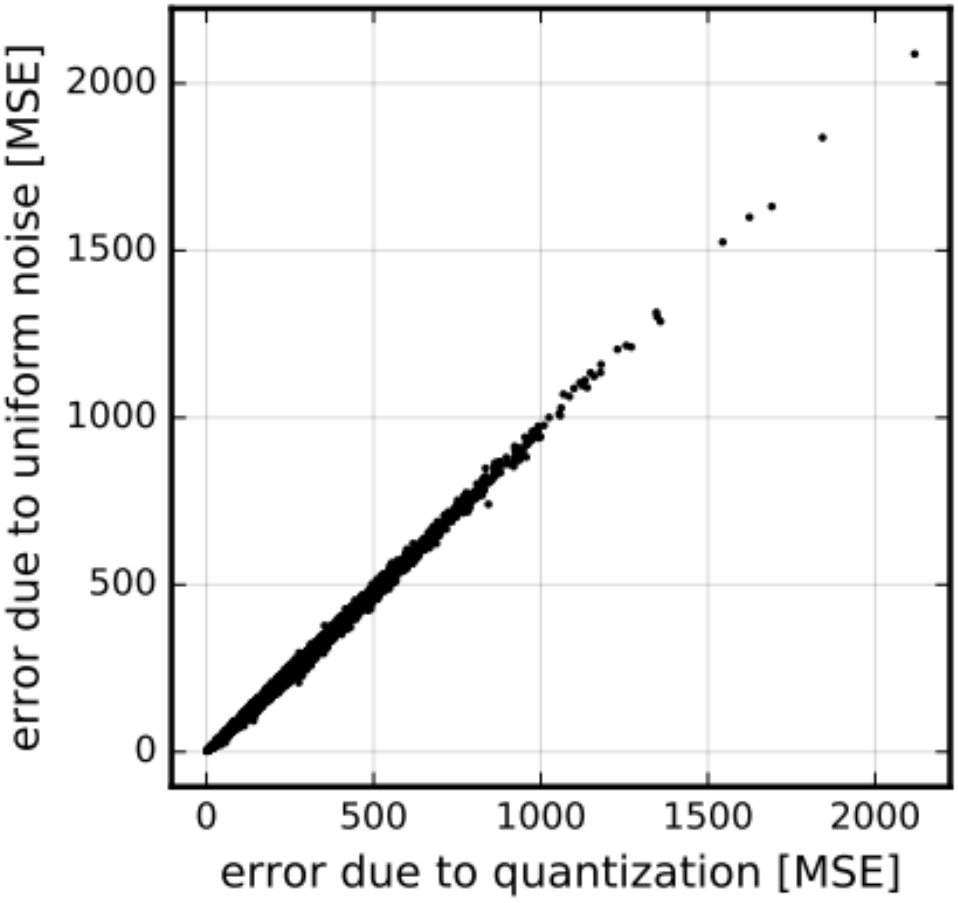
At train time (but not test time) the rounding is replaced with additive noise.

$$\Phi^* = \underset{\Phi}{\text{argmin}}\ E_{y,\epsilon} - \log_2 P_\Phi(C_\Phi(y) + \epsilon) + \lambda||y - \hat{y}_\Phi(C_\Phi(y) + \epsilon)||^2$$

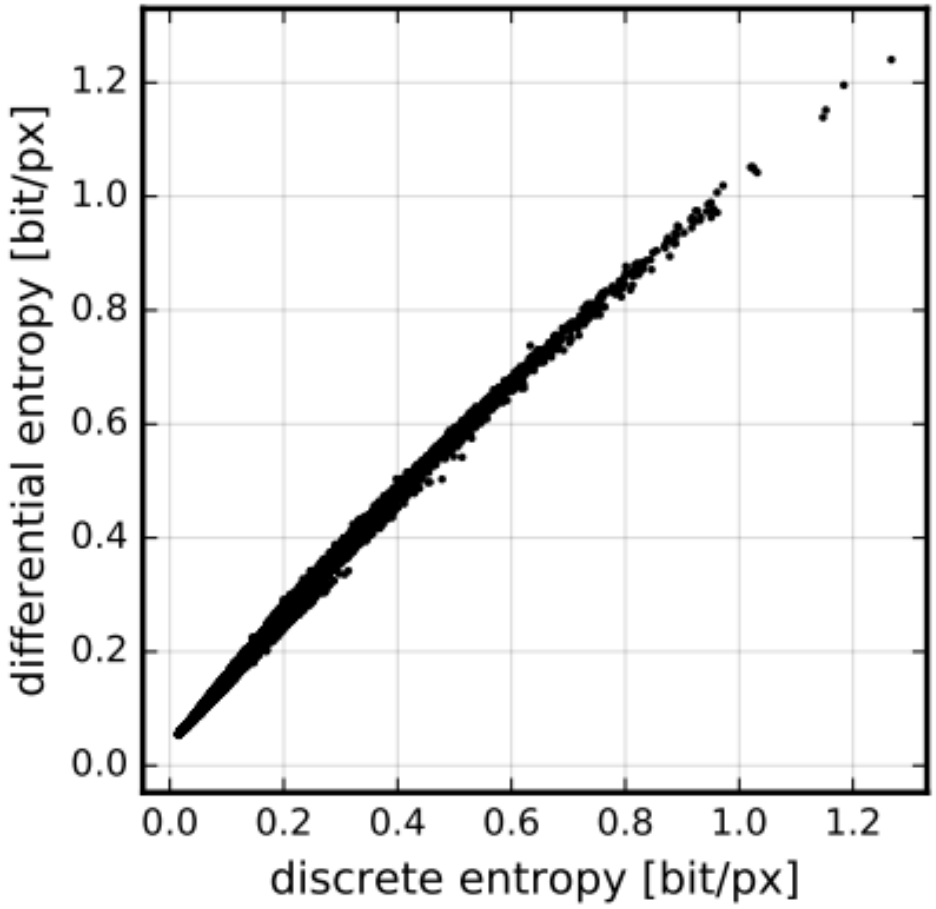
$$\epsilon_i \text{ drawn uniformly from } [-1/2,\ 1/2]$$

P_Φ defines a piecewise linear density for each coordinate of z.

Noise vs. Rounding



Differential Entropy vs. Discrete Entropy



Varying the Level Of Compression

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}}\ E_{y\sim \mathrm{Pop}} -\log_2 P_\Phi(\tilde{C}_\Phi(y)) + \frac{1}{2}\lambda ||y - \hat{y}_\Phi(\tilde{C}_\Phi(y))||^2$$

Different levels of compression correspond to different values of λ.

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

Higher levels of compression result in more compact distributions on the 192 numbers.

END