TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2018

Autoencoders:

Rate-Distortion Autoencoders

Variational Autoencoders

PS3 Winners

Accuracy over four epochs.

Each person scored in the top three (allowing for ties) in at least one epoch.

Ruizhe Zhou	52.14	53.33	59.97	63.50
Chen Zou	54.77	62.12	65.30	66.15
Shuo Zhang	46.53	51.03	54.04	66.39
Mark Vandergon	43.46	66.36	66.79	66.47
Sidharth Sachdeva	49.81	60.81	61.83	70.04
Zi Ye	40.87	47.00	64.84	67.49
Heqing Ye	50.44	55.41	61.30	64.05
Nyanyi Chen	44.56	61.50	63.74	63.56
Shenqi Sang	42.98	60.34	65.52	

$\eta = B\eta_{1,0}(1-\mu)$??

Rate-Distortion Autoencoders (Data Compression)

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \quad E_{(x,y)\sim \mathrm{Pop}} - \log Q_\Phi(y|x)$$

Rate-distortion autoencoders (data compression algorithms) have measurable performance.

Data Compression

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \quad E_{(x,y)\sim \mathrm{Pop}} - \log Q_\Phi(y|x)$$

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \quad E_{(x,y)\sim \mathrm{Pop}} \quad |C_\Phi(y|x)| - \log_2 Q_\Phi(y \mid \hat{y}_\Phi(x, C_\Phi(y|x)))$$

Rate Distortion

Data Compression

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \quad E_{(x,y)\sim \mathrm{Pop}} - \log Q_\Phi(y|x)$$

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \quad E_{(x,y)\sim \mathrm{Pop}} \quad |C_\Phi(y|x)| - \log_2 Q_\Phi(y \mid \hat{y}_\Phi(C_\Phi(y|x)))$$

$$Q_\Phi(y|\hat{y}) \propto \exp\left(\frac{-||y-\hat{y}||^2}{2\sigma_2}\right)$$

$$\textbf{Distortion} = -\log_2 Q_\Phi(y|\hat{y}) = \frac{1}{2\sigma^2}||y-\hat{y}||^2 + g(\sigma)$$

for some function g.

Data Compression

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \quad E_{(x,y)\sim \mathrm{Pop}} - \log Q_\Phi(y|x)$$

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \quad E_{(x,y)\sim \mathrm{Pop}} \quad |C_\Phi(y|x)| - \log_2 Q_\Phi(y \mid \hat{y}_\Phi(x, C_\Phi(y|x)))$$

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

$$\lambda = \frac{1}{\sigma^2}$$

Data Compression

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

If the image can be segmented based on x then $C_\Phi(y|x)$ can be the specification of the light temperature and the intrinsic color of each segment.

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.

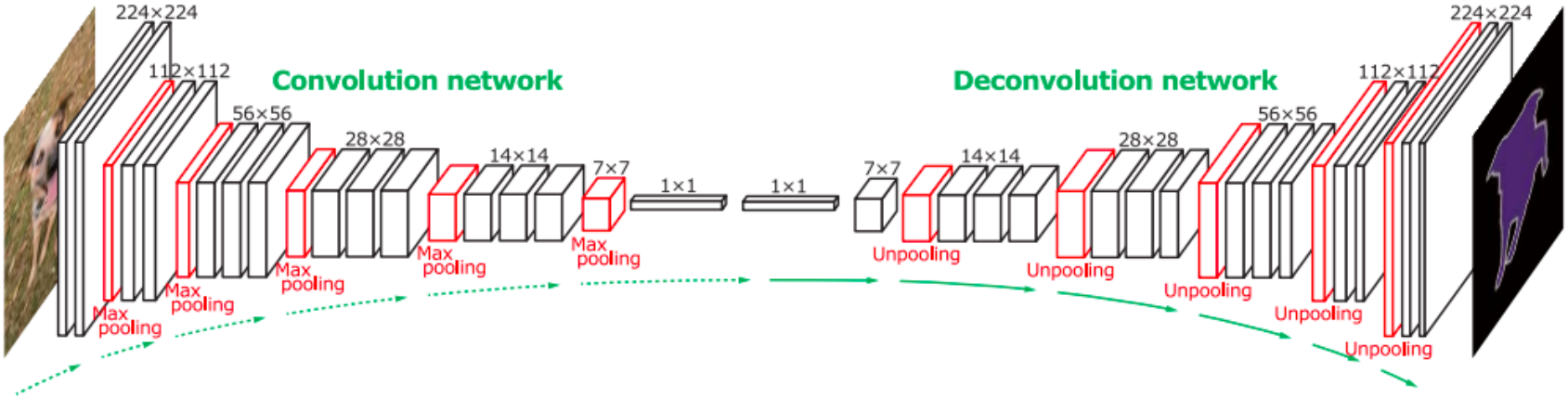
Final image dimension is reduced by a factor of 16 with 192 channels per pixel (192 channels is for color images).

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

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

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

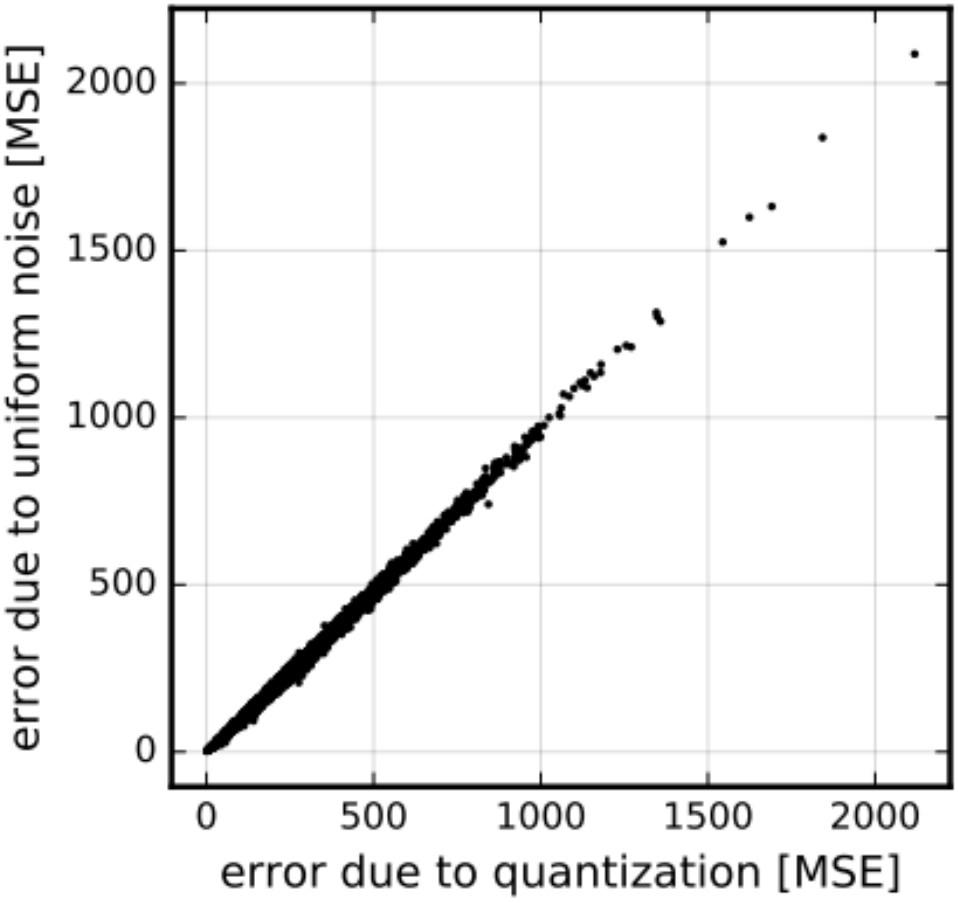
At train time (but not test time) the rounding is replaced with additive noise.

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

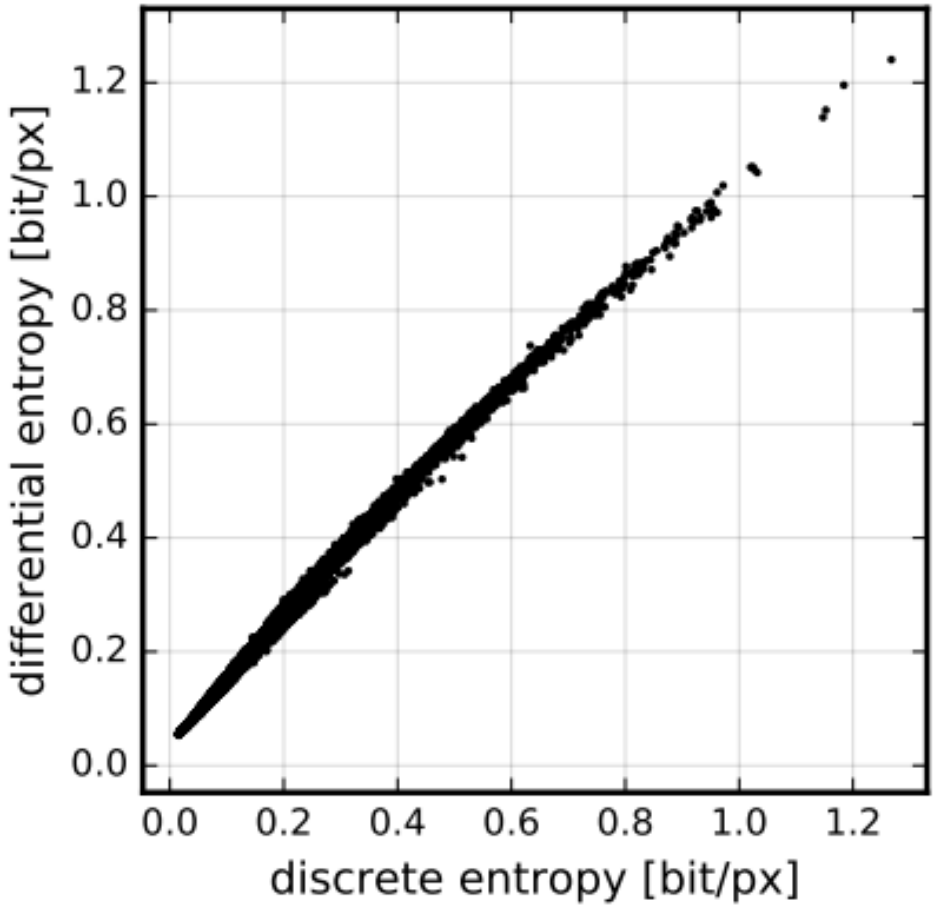
$$\epsilon_i \text{ drawn uniformly from } [-1/2,\ 1/2]$$

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.

Variational Autoencoders

Motivation: Fundamental Equations for Latent Variables

We will now drop the negation and switch to argmax.

$$\Phi^* = \operatorname*{argmax}_{\Phi} \; E_{y \sim \mathrm{Pop}} \ln Q_\Phi(y)$$

$$Q_\Phi(y) = \sum_{\hat{z}} Q_\Phi(\hat{z}, y)$$

EG Identity: $$\nabla_\Phi \ln Q_\Phi(y) = \mathrm{E}_{\hat{z} \sim Q_\Phi(\hat{z}|y)} \; \nabla_\Phi \; \ln Q_\Phi(\hat{z}, y)$$

A #P-Hard Problem

$$\nabla_\Phi \ln Q_\Phi(y) = \mathrm{E}_{\hat{z} \sim Q_\Phi(\hat{z}|y)} \nabla_\Phi \ \ln Q_\Phi(\hat{z}, y)$$

For a latent variable graphical model we can compute $\nabla_\Phi \ln Q_\Phi(y)$ from a small number of sufficient statistics summarizing the distribution $Q_\Phi(\hat{z}|y)$.

However, except for tree-structured directed graphical models (such as CTC), computing the sufficient statistics of $Q_\Phi(\hat{z}|y)$ is #P-hard.

Variational Autoencoders

$$\nabla_\Phi \ln Q_\Phi(y) = \mathrm{E}_{\hat{z}\sim Q_\Phi(\hat{z}|y)} \nabla_\Phi \ln Q_\Phi(\hat{z}, y)$$

$$\Phi^*, \Psi^* = \underset{\Phi,\Psi}{\operatorname{argmax}} \mathrm{E}_{\hat{z}\sim P_\Psi(\hat{z}|y)} \ln Q_\Phi(\hat{z}, y) + H(P_\Psi(\hat{z}|y))$$

Variation autoencoders replace the expectation over $Q_\Phi(\hat{z}|y)$ with an expectation over another model $P_\Psi(\hat{z}|y)$.

Variational Autoencoders

$$\nabla_\Phi \ln Q_\Phi(y) = \mathrm{E}_{\hat{z}\sim Q_\Phi(\hat{z}|y)} \nabla_\Phi \ln Q_\Phi(\hat{z}, y)$$

$$\Phi^*, \Psi^* = \underset{\Phi,\Psi}{\operatorname{argmin}} \mathrm{E}_{\hat{z}\sim P_\Psi(\hat{z}|y)} \ln Q_\Phi(\hat{z}, y) + H(P_\Psi(\hat{z}|y))$$

We assume that calculating $Q_\Phi(\hat{z}, y)$ is easy but sampling from $Q_\Phi(\hat{z}|y)$ is hard.

We require that sampling from $P_\Psi(\hat{z}|y)$ and evaluating $P_\Psi(\hat{z}|y)$ are both easy.

Variational Autoencoders

$$\nabla_\Phi \ln Q_\Phi(y) = \mathrm{E}_{\hat{z} \sim Q_\Phi(\hat{z}|y)} \nabla_\Phi \ \ln Q_\Phi(\hat{z}, y)$$

$$\Phi^*, \Psi^* = \underset{\Phi,\Psi}{\operatorname{argmax}} \ \mathrm{E}_{\hat{z} \sim P_\Psi(\hat{z}|y)} \ \ln Q_\Phi(\hat{z}, y) + H(P_\Psi(\hat{z}|y))$$

$Q_\Phi(\hat{z}, y)$ can be a directed (locally normalized) loopy model.

$P_\Psi(\hat{z}|y)$ can be a tree model.

Variational Autoencoders

$$\nabla_\Phi \ln Q_\Phi(y) = \mathrm{E}_{\hat{z}\sim Q_\Phi(\hat{z}|y)} \nabla_\Phi \ln Q_\Phi(\hat{z}, y)$$

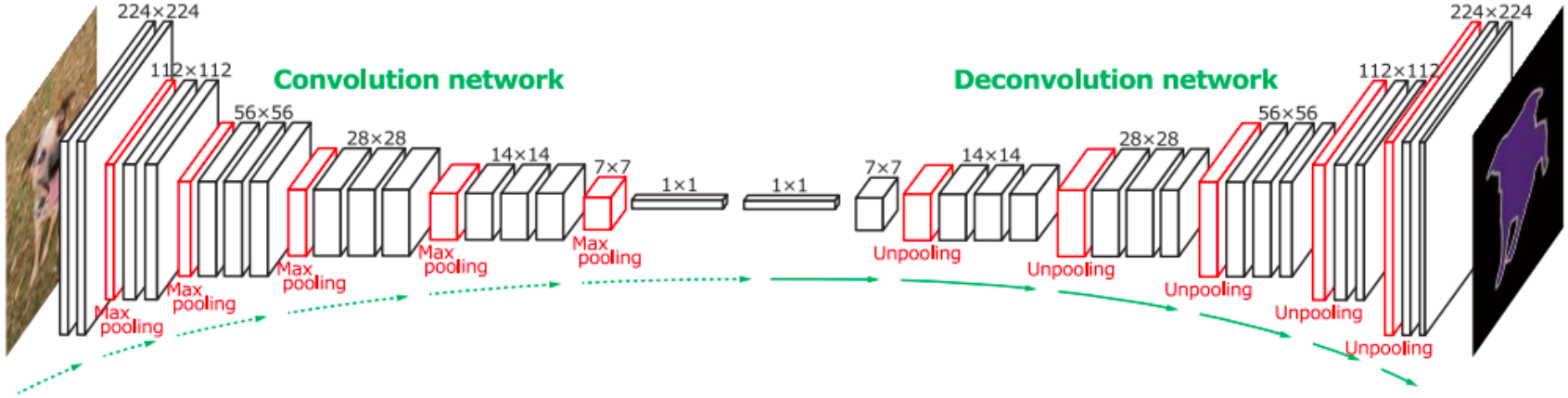
$$\Phi^*, \Psi^* = \underset{\Phi,\Psi}{\operatorname{argmax}} \mathrm{E}_{\hat{z}\sim P_\Psi(\hat{z}|y)} \ln Q_\Phi(\hat{z}, y) + H(P_\Psi(\hat{z}|y))$$

$Q_\Phi(\hat{z}, y)$ can be defined by drawing $\hat{z}$ from a Gaussian, appyling deconvolution, and adding Gaussian noise to each resulting pixel.

$P_\Psi(\hat{z}|y)$ can be defined by a CNN computing parameters of a Gaussian.

The Gaussian Case

$$\Phi^*, \Psi^* = \underset{\Phi,\Psi}{\operatorname{argmax}} \; \mathrm{E}_{\hat{z}\sim P_\Psi(\hat{z}|y)} \; \ln Q_\Phi(\hat{z}, y) + H(P_\Psi(\hat{z}|y))$$



[Hyeonwoo Noh et al.]

The Theorem: The Evidence Lower Bound (ELBO)

$$\begin{aligned}
\text{EG: } \nabla_\Phi \ln Q_\Phi(y) &= \mathrm{E}_{\hat{z}\sim Q_\Phi(\hat{z}|y)} \nabla_\Phi \ln Q_\Phi(\hat{z}, y) \\
\text{ELBO: } \mathcal{L}(\Phi, \Psi, y) &\doteq \mathrm{E}_{\hat{z}\sim P_\Psi(\hat{z}|y)} \ln Q_\Phi(\hat{z}, y) + H(P_\Psi(\hat{z}|y)) \\
&= \ln Q_\Phi(y) - KL(P_\Psi(\hat{z}|y), Q_\Phi(\hat{z}|y)) \\
\text{Evidence: } &\leq \ln Q_\Phi(y)
\end{aligned}$$

Proof I

$$
\begin{aligned}
& E_{\hat{z}\sim P_\Psi(\hat{z}|y)} \ \ln Q_\Phi(\hat{z}, y) \\
= \ & E_{\hat{z}\sim P_\Psi(\hat{z}|y)} \ (\ \ln Q_\Phi(y) + \ln Q_\Phi(\hat{z}|y) \ ) \\
= \ & \ln Q_\Phi(y) + \ E_{\hat{z}\sim P_\Psi(\hat{z}|y)} \ \ln Q_\Phi(\hat{z}|y) \\
= \ & \ln Q_\Phi(y) - H(P_\Psi(\hat{z}|y), Q_\Phi(\hat{z}|y))
\end{aligned}
$$

Proof II

$$\begin{aligned}
\mathcal{L}(\Phi, \Psi, y) &\doteq \mathrm{E}_{\hat{z} \sim P_\Psi(\hat{z}|y)} \ln Q_\Phi(\hat{z}, y) + H(P_\Psi(\hat{z}|y)) \\
&= \ln Q_\Phi(y) - H(P_\Psi(\hat{z}|y), Q_\Phi(\hat{z}|y)) + H(P_\Psi(\hat{z}|y)) \\
&= \ln Q_\Phi(y) - KL(P_\Psi(\hat{z}|y), Q_\Phi(\hat{z}|y))
\end{aligned}$$

EM: Alternating Optimization of the ELBO

$$\mathcal{L}(\Phi, \Psi, y) = E_{\hat{z} \sim P_\Psi(\hat{z}|y)} \ln Q_\Phi(\hat{z}, y) + H(P_\Psi(\hat{z}|y)) \quad (1)$$

$$= \ln Q_\Phi(y) - KL(P_\Psi(\hat{z}|y), Q_\Phi(\hat{z}|y)) \quad (2)$$

$$\text{by (2)} \quad \Psi^* = \underset{\Psi}{\operatorname{argmax}} KL(P_\Psi(\hat{z}|y), Q_\Phi(\hat{z}|y))$$

$$\text{by (1)} \quad \Phi^* = \underset{\Phi}{\operatorname{argmax}} E_{\hat{z} \sim P_\Psi(\hat{z}|y)} \ln Q_\Phi(\hat{z}, y)$$

As a single update:

$$\Phi^{t+1} = \underset{\Phi}{\operatorname{argmax}} \quad E_{\hat{z} \sim Q_{\Phi^t}(\hat{z}|y)} \quad \log Q_\Phi(\hat{z}, y)$$

The Reparameterization Trick

$$\nabla_\Psi \, \mathcal{L}(\Phi, \Psi, y)$$

$$= \nabla_\Psi \, \mathrm{E}_{\hat{z} \sim P_\Psi(\hat{z}|y)} \, \ln Q_\Phi(\hat{z}, y) + H(P_\Psi(\hat{z}|y))$$

The sampling depends on Ψ. How do we differentiate the sampling?

The Reparameterization Trick

Model $P_\Psi(\hat{z}|y)$ by $\epsilon \sim$ noise, $\hat{z} = \hat{z}_\Psi(y, \epsilon)$

$$\nabla_\Psi \mathcal{L}(\Phi, \Psi, y)$$

$$= \nabla_\Psi \, \mathrm{E}_{\hat{z} \sim P_\Psi(\hat{z}|y)} \, \ln Q_\Phi(\hat{z}, y) + H(P_\Psi(\hat{z}|y))$$

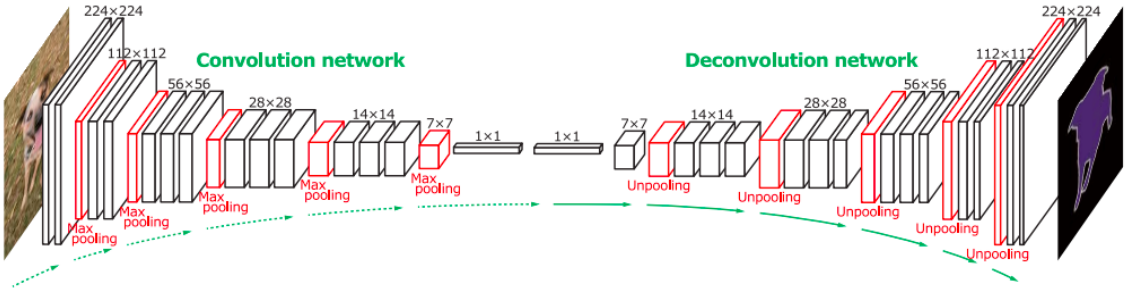
$$= \nabla_\Psi \, \mathrm{E}_{\epsilon \sim \mathrm{Noise}} \, \ln Q_\Phi(\hat{z}_\Psi(y, \epsilon), y) + H(P_\Psi(\hat{z}|y))$$

$z \sim \mathcal{N}(\mu, \sigma)$

becomes $\epsilon \sim \mathcal{N}(0, 1)$ $\hat{z} = \mu + \sigma\epsilon.$

Sampling

$$P_\Psi(\hat{z}|y) \quad \hat{z} \quad Q_\Phi(\hat{z}, y)$$



[Hyeonwoo Noh et al.]

Sampling uses just the second half $Q_\Phi(\hat{z}, y)$.

Sampling from Gaussian Variational Autoencoders

[Alec Radford]

END