# Problem Set 10

*published: 28 June 2023*
*discussion: 5 July 2023*

**Data Compression With And Without Deep Probabilistic Models**
Prof. Robert Bamler, University of Tübingen

Course materials available at https://robamler.github.io/teaching/compress23/

## Problem 10.1: Beta Variational Autoencoder ($\beta$-VAE)

Consider a variational autoencoder (VAE) with a Gaussian likelihood $P_{\theta,\beta}(\mathbf{X}\,|\,\mathbf{Z}{=}\mathbf{z}) = \mathcal{N}(g_\theta(\mathbf{z}), \frac{\beta}{2}I)$.[1] Here, the mean $g_\theta(\mathbf{z})$ of the normal distribution is the output of a neural network $g_\theta$ with (learnable) weights $\theta$, and the scalar hyperparameter $\beta > 0$ is held fixed during training. We denote the variational distribution by $Q_\phi(\mathbf{Z}\,|\,\mathbf{X})$ as usual, where $\phi$ are the (learnable) weights of an encoder network (aka "inference network") $f_\phi$.

(a) For now, we don't make any assumption about the variational family. Show that maximizing $\mathbb{E}_{\mathbf{x}\sim\text{training set}}\Big[\text{ELBO}(\theta,\phi,\mathbf{x})\Big]$ over the neural network weights $\phi$ and $\theta$ is equivalent to minimizing a rate/distortion trade-off

$$\mathcal{L}_\beta(\theta,\phi) := \mathbb{E}_{\mathbf{x}\sim\text{training set}}\Big[\mathcal{D}(\theta,\phi,\mathbf{x}) + \beta\,\mathcal{R}(\theta,\phi,\mathbf{x})\Big] \tag{1}$$

with the

- distortion $\mathcal{D}(\theta,\phi,\mathbf{x}) := \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{Z}|\mathbf{X}=\mathbf{x})}\Big[\|g_\theta(\mathbf{z}) - \mathbf{x}\|_2^2\Big]$; and the
- rate $\mathcal{R}(\theta,\phi,\mathbf{x}) := D_{\text{KL}}\Big(Q_\phi(\mathbf{Z}\,|\,\mathbf{X}{=}\mathbf{x})\,\Big\|\,P_\theta(\mathbf{Z})\Big)$.

(b) We now consider a box-shaped variational distribution with fixed width 1, i.e.,

$$Q_\phi(\mathbf{Z}\,|\,\mathbf{X}{=}\mathbf{x}) = \prod_{i=1}^{d} \mathcal{U}\Big(Z_i; \Big[f_\phi(\mathbf{x})_i - \tfrac{1}{2}, f_\phi(\mathbf{x})_i + \tfrac{1}{2}\Big]\Big) \tag{2}$$

where $d$ is the dimensionality of $\mathbf{Z}$-space, and $\mathcal{U}(Z_i; [a,b])$ denotes that the vector component $Z_i$ is uniformly distributed over the interval $[a,b]$ (i.e., its probability density function (pdf) $q_{\phi,i}(z_i\,|\,\mathbf{X}{=}\mathbf{x})$ has a constant positive value for $z_i \in [a,b]$ and is zero for $z_i \notin [a,b]$). Denote the pdf of the prior $P_\theta(\mathbf{Z})$ by $p_\theta$ and show that

$$\mathcal{R}(\theta,\phi,\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{Z}|\mathbf{X}=\mathbf{x})}\Big[-\log p_\theta(\mathbf{z})\Big] = \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{Z}|\mathbf{X}=\mathbf{x})}\left[-\sum_{i=1}^{d}\log p_{\theta,i}(z_i)\right] \tag{3}$$

where the last equality assumes that the prior $P_\theta(\mathbf{Z}) = \prod_{i=1}^d P_\theta(Z_i)$ is fully factorized with a pdf $p_{\theta,i}$ for each component $z_i$.

[1] Early versions of the lecture notes had the covariance matrix mistakenly as $\frac{1}{\beta}I$ instead of $\frac{\beta}{2}I$.

(c) Let's now fully embrace the rate/distortion interpretation of the loss $\mathcal{L}_\beta(\theta, \phi)$. Thus, we abandon the probabilistic interpretation of the encoder and decoder model and focus instead on the deterministic functions $f_\phi$ and $g_\theta$ that map from image space to latent space and vice versa, respectively. Our goal is to eventually use the trained model for lossy data compression.

*In deployment*, we round each component $z_i$ of $\mathbf{z} := f_\phi(\mathbf{x})$ to the nearest integer, $\hat{z}_i := \lceil z_i \rfloor \in \mathbb{Z}$. We then entropy code each $\hat{z}_i$ using the probability mass function $P(\hat{Z}_i = \hat{z}_i) = \int_{\hat{z}_i - \frac{1}{2}}^{\hat{z}_i + \frac{1}{2}} p_{\theta,i}(z_i)\,\mathrm{d}z_i$, thus obtaining a bit rate of $-\log P(\hat{Z}_i = \hat{z}_i)$.

*During training*, we replace the nondifferentiable rounding operation by sampling from the box-shaped distribution in Eq. 2 (Ballé et al., 2017): $\mathbf{z} \sim Q_\phi(\mathbf{Z} \,|\, \mathbf{X} = \mathbf{x})$. Everything else should stay as close as possible to the situation during deployment. Therefore, we replace the pdf $p_{\theta,i}(z_i)$ on the right-hand side of Eq. 3 with

$$\tilde{p}_{\theta,i}(z_i) := \int_{z_i - \frac{1}{2}}^{z_i + \frac{1}{2}} p_{\theta,i}(z_i')\,\mathrm{d}z_i' \tag{4}$$

so that the resulting rate term $\mathcal{R}(\theta, \phi, \mathbf{x})$ resembles as closely as possible the bit rate that we obtain when we entropy code with the model $P(\hat{Z}_i = \hat{z}_i) = \int_{\hat{z}_i - \frac{1}{2}}^{\hat{z}_i + \frac{1}{2}} p_{\theta,i}(z_i)\,\mathrm{d}z_i$ during deployment.

Assume that

$$p_{\theta,i}(z_i) = \mathcal{N}(z_i; 0, \sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma_i^2}}\, e^{-z_i^2/(2\sigma_i^2)} \tag{5}$$

is a normal distribution with zero mean and (learnable) variance $\sigma_i^2$. Show that

$$\tilde{p}_{\theta,i}(z_i) = \frac{1}{2}\left[\operatorname{erf}\left(\frac{z_i + \frac{1}{2}}{\sqrt{2}\,\sigma_i}\right) - \operatorname{erf}\left(\frac{z_i - \frac{1}{2}}{\sqrt{2}\,\sigma_i}\right)\right] \tag{6}$$

where the error function is defined as

$$\operatorname{erf}(\xi) := \frac{2}{\sqrt{\pi}} \int_0^\xi e^{-t^2}\,\mathrm{d}t. \tag{7}$$

## Problem 10.2: Lossy Compression With a $\beta$-VAE

The accompanying jupyter notebook implements a toy beta variational autoencoder ($\beta$-VAE) that can be used for lossy data compression. It is very similar to the VAE implementation that we looked at last week for lossless compression. But I removed all parts that were specific to lossless compression. Your task is to replace these parts with the appropriate code to implement the $\beta$-VAE from Problem 10.1 (b) above.

Search for "TODO" in the notebook to find all places where you have to do something.

(a) `EntropyBottleneck`: we use a prior $P_\theta(\mathbf{Z})$ that is a normal distribution with mean zero and diagonal covariance matrix whose components are learned. To avoid committing to a specific image size at training time, our encoder and decoder networks are fully (de)convolutional, and the learned standard deviations of the prior are the same at every spacial position (but different for every channel, i.e., last index of $\mathbf{Z}$). Most of this is already implemented for you, but there are about two missing lines of code in the method `forward` where you need to fill in the implementation of $\tilde{p}_{\theta,i}(z_i)$ from Eq. 6.

(b) `EncoderModel`: since our box-shaped variational distribution (Eq. 2) has a fixed width, the encoder network only outputs its mean `q_mean` $:= f_\phi(\mathbf{x})$. This is already implemented for you in the method `forward`. Implement the method `reparameterize`, which draws a sample from $Q_\phi(\mathbf{Z} \mid \mathbf{X}{=}\mathbf{x})$ (see hints in the code).

(c) `DecoderModel`: implement the method `distortion`, which estimates $\mathcal{D}(\theta, \phi, \mathbf{x})$ from Problem 10.1 (a). Here, the method argument `reconstruction` is $g_\phi(\mathbf{z})$.

(d) `bit_rate_and_reconstruction`: this method executes a round trip from (a batch of) original images $\mathbf{x}$ to latent representations $\mathbf{z}$ to reconstrutions $\mathbf{x}'$. Most of this is already implemented for you. But one line of code is missing, and there is also one line of code that I removed from the VAE implementation in the last problem set (see comment in the code). Fill in the missing line of code and explain why the removed line of code does not apply to our interpretation of lossy compression.

(e) *Rate/distortion curves:* all the boilerplate code for the training loop is already implemented for you. Train the VAE once with $\beta = 0.1$ and `z_channels` $= 2$ as already indicated in the code and verify that all reported values are plausible (ask yourself what bit rates or distortions would be implausible; what baseline can you compare to?). Then try out various values for $\beta$ and `z_channels` and plot rate/distortion curves as indicated in the notebook.

(f) *Deployment in a lossy compression pipeline:* Once you've trained a model with reasonable settings for $\beta$ and `z_channels`, you can use it for lossy data compression. The function `encode_single_image` is already implemented for you. Implement the function `decode_single_image` by following the instructions in the comment. Then run `test_compression` as indicated in the cells below and compare the empirical bit rates and distortions to the estimates that you obtained during training. Can you explain why there are some small discrepancies?

## Problem 10.3: Data Processing Inequality

In this problem, you will prove a fundamental theorem of communication systems: the data processing inequality. This inequality will become crucial for the theory of lossy compression, and for channel coding theory.

Consider a *Markov chain* $X_1 \to X_2 \to \cdots \to X_n$ (see Lecture 4 and Problem Set 5), i.e., random variables $X_1, \ldots, X_n$ whose joint distribution factorizes as follows,

$$P(X_1, X_2, \ldots, X_n) = P(X_1)\, P(X_2|X_1)\, P(X_3|X_2) \cdots P(X_n|X_{n-1}) \\ \text{(for a Markov chain } X_1 \to X_2 \to \cdots \to X_n\text{)}. \tag{8}$$

As discussed on Problem Set 5, a Markov chain models a *memoryless* process, i.e., a chain of stateless stochastic operations where each stochastic operation takes as input *only* the output of the immediately preceding operation. For example, think about kids at a birthday party who play a game of telegraph (German: "Flüsterpost").

The data processing inequality makes two statements about how information propagates through such a Markov chain:

- Information about a fixed *past* $i$ can only *decrease* along a Markov chain:

$$I_P(X_i; X_j) \geq I_P(X_i; X_k) \qquad \forall\, i < j < k \qquad \text{(for Markov chains)}. \tag{9}$$

- Information about a fixed *future* $k$ can only *increase* along a Markov chain:

$$I_P(X_i; X_k) \leq I_P(X_j; X_k) \qquad \forall\, i < j < k \qquad \text{(for Markov chains)}. \tag{10}$$

.

The following steps guide you through the proofs of Eqs. 9 and 10.

(a) In order to relate Eqs. 9-10 to their respective verbal statements, recall why the mutual information $I_P(X;Y)$ can be interpreted as a measure of how much information $Y$ gives us about $X$ and vice versa. This was discussed in Problem 4.4 (c) on Problem Set 4 and in the paragraph marked "Interpretation" below it.

(b) Consider a Markov chain of length $n \geq 3$. Recall that, if we pick three items $X_i$, $X_j$, and $X_k$ in order (i.e., with $i < j < k$), then they form a Markov chain $X_i \to X_j \to X_k$ (if this is not obvious to you, then refer back to Problem 5.2 (b) (ii) on Problem Set 5). Thus, $P(X_i, X_j, X_k) = P(X_i)P(X_j|X_i)P(X_k|X_j)$.

(c) Use $I_P(X;Y) = H_P(Y) - H_P(Y|X)$ (see Problem 4.4 (c)) to derive a relation

$$I_P(X_j; X_k) - I_P(X_i; X_k) = \mathbb{E}_P\left[ -\log_2 \left( \frac{P(X_?|X_?)}{P(X_?|X_?)} \right) \right] \tag{11}$$

where each "?" is either $i$, $j$, or $k$.

(d) Use Jensen's inequality (see Problem 3.1 (b)) to pull the logarithm in Eq. 11 out of the expectation. Then write out the expectation as a weighted average (using the fact that $P(X_i, X_j, X_k) = P(X_i)P(X_j|X_i)P(X_k|X_j)$) and prove Eq. 10.

(e) To prove Eq. 9, recall that the statement "$X_i \to X_j \to X_k$ is a Markov chain" is equivalent to the statement "$X_i$ and $X_k$ are conditionally independent given $X_j$" (see Problem 5.1 (a)). Use the symmetry of conditional independence to argue that $X_k \to X_j \to X_i$ is then also a Markov chain and therefore Eq. 9 holds.

(f) **What is information?** The data processing inequality can be interpreted as follows: assume we feed some input data $X_1$ into some (possibly nondeterministic) machine that processes the data and outputs $X_2$, and we then feed $X_2$ (but not $X_1$) into some other (possibly nondeterministic) machine that outputs $X_3$. Using the interpretation of the mutual information reviewed in part (a), the data processing inequality Eq. 9 then tells us that any information about $X_1$ that gets destroyed by the first machine cannot be regenerated by the second machine.

Think about what this means for the interpretation of our notion of "information". How well does our formal notion of the "information content" capture what we would colloquially consider as "information"? For example, think about a cryptographic pipeline $X_1 \to X_2 \to X_3$ where $X_1$ is a clear text message, $X_2$ is the encrypted representation of $X_1$, and $X_3$ is the decrypted message (thus, $X_3 = X_1$). What does Eq. 9 imply about $I_P(X_1; X_2)$?

Or think about a crime scene, where the perpetrator first destroys as much evidence as they can, and the police then recover some of it. How much information about the crime do the police unveil, according to our very specific notion of information?

These considerations should be a reminder that information theory uses a very specific notion of the term "information". In particular, this notion of information does not take into account a computational model or other physical constraints, which is a crucial difference between information theory and cryptography.

The term "information" is used in information theory as a *metaphor*. In the same way in which no physicist would claim that their notion of the term "power" (energy transferred per time) bears any meaning about the notion of "power" in a political context, we should also never forget that technical terms like "information" (or also "intelligence") are used in information theory (or artificial intelligence) only as metaphors, and that this specific choice of words is, to some degree, a meaningless historical coincidence.

# References

Ballé, J., Laparra, V., and Simoncelli, E. P. (2017). End-to-end optimized image compression. *International Conference on Learning Representations*.