Coming back to generating a signal, instead of training an autoencoder and modeling the distribution of $Z$, we can try an alternative approach:

**Impose a distribution for** $Z$ and then train a decoder $g$ so that $g(Z)$ matches the training data.
We consider the following two distributions:

- $p$ is the distribution on $\mathcal{X} \times \mathbb{R}^d$ of a pair $(X, Z)$ composed of an encoding state $Z \sim \mathcal{N}(0, I)$ and the output of the decoder $g$ on it.

- $q$ is the distribution on $\mathcal{X} \times \mathbb{R}^d$ of a pair $(X, Z)$ composed of a sample $X$ taken from the data distribution and the output of the encoder on it.

Our goal is that $p(X)$ mimics the data-distribution $q(X)$, that is to find $g$ that maximizes the log-likelihood

$$\frac{1}{N} \sum_n \log p(x_n) = \hat{E}_q(X) \left[ \log p(X) \right].$$

However, with a complicated $g$, we can sample $z$ and compute $g(z)$, but cannot compute $p(x)$ for a given $x$, and even less compute its derivatives.

The Variational Autoencoder proposed by Kingma and Welling (2013) relies on a tractable approximation of this log-likelihood.

Note that their framework involves stochastic encoder $f$, and decoder $g$, whose outputs depend on both their inputs and additional randomness.
Remember that \( q(X) \) is the data distribution, and \( q(Z \mid X = x) \) is the distribution of the latent encoding \( f(x) \). We want to maximize

\[
\mathbb{E}_{q(X)} \left[ \log p(X) \right],
\]

and we can show that

\[
- \mathbb{E}_{q(X)} \left[ \log p(X) \right] \leq \mathbb{E}_{q(X)} \left[ D_{KL}(q(Z \mid X) \parallel p(Z)) \right] - \mathbb{E}_{q(X,Z)} \left[ \log p(X \mid Z) \right].
\]

So it makes sense to minimize this latter quantity.

So the final loss is

\[
\mathcal{L} = \mathbb{E}_{q(X)} \left[ D_{KL}(q(Z \mid X) \parallel p(Z)) \right] - \mathbb{E}_{q(X,Z)} \left[ \log p(X \mid Z) \right].
\]

with

- \( q(X) \) is the data distribution
- \( p(Z) = \mathcal{N}(0, I) \).

Kingma and Welling propose that both the encoder \( f \) and decoder \( g \) map to a Gaussian with diagonal covariance. Hence they map to twice the dimension (e.g. \( f(x) = (\mu_f(x), \sigma_f(x)) \)) and

- \( q(Z \mid X = x) \sim \mathcal{N}(\mu_f(x), \text{diag}(\sigma_f(x))) \)
- \( p(X \mid Z = z) \sim \mathcal{N}(\mu_g(z), \text{diag}(\sigma_g(z))) \).
The first term of $\mathcal{L}$ is the average of

$$
D_{KL} \left( q(Z \mid X = x) \parallel p(Z) \right) = -\frac{1}{2} \sum d \left( 1 + 2 \log \sigma_f^d(x) - \left( \mu_f^d(x) \right)^2 - \left( \sigma_f^d(x) \right)^2 \right).
$$

over the $x_n$s.

This can be implemented as

```python
param_f = model.encode(input)
mu_f, logvar_f = param_f.split(param_f.size(1)//2, 1)
kl = - 0.5 * (1 + logvar_f - mu_f.pow(2) - logvar_f.exp())
kl_loss = kl.sum() / input.size(0)
```
As Kingma and Welling (2013), we use a constant variance of 1 for the decoder, so the second term of $\mathcal{L}$ becomes the average of

$$-\log p(X = x \mid Z = z) = \frac{1}{2} \sum_d (x_d - \mu^g_d(z))^2 + \text{cst}$$

over the $x_n$, with one $z_n$ sampled for each, i.e.

$$z_n \sim \mathcal{N} \left( \mu^f(x_n), \sigma^f(x_n) \right), \ n = 1, \ldots, N.$$

This can be implemented as

```python
std_f = torch.exp(0.5 * logvar_f)
z = torch.empty_like(mu_f).normal_() * std_f + mu_f
output = model.decode(z)
fit = 0.5 * (output - input).pow(2)
fit_loss = fit.sum() / input.size(0)
```
We had for the standard autoencoder

```
z = model.encode(input)
output = model.decode(z)
loss = 0.5 * (output - input).pow(2).sum() / input.size(0)
```

and putting everything together we get for the VAE

```
param_f = model.encode(input)
mu_f, logvar_f = param_f.split(param_f.size(1)//2, 1)
kl = - 0.5 * (1 + logvar_f - mu_f.pow(2) - logvar_f.exp())
kl_loss = kl.sum() / input.size(0)
std_f = torch.exp(0.5 * logvar_f)
z = torch.empty_like(mu_f).normal_() * std_f + mu_f
output = model.decode(z)

fit = 0.5 * (output - input).pow(2)
fit_loss = fit.sum() / input.size(0)
loss = kl_loss + fit_loss
```

During inference we do not sample, and instead use $\mu^f$ and $\mu^g$ as prediction.

Note in particular the re-parameterization trick:

```
z = torch.empty_like(mu_f).normal_() * std_f + mu_f
output = model.decode(z)
```

Implementing the sampling of $z$ that way allows to compute the gradient w.r.t $f$’s parameters without any particular property of `normal_()`.
We can look at two latent features to check that they are Normal for the VAE.
Autoencoder sampling ($d = 32$)

Variational Autoencoder sampling ($d = 32$)

Making the embedding $\sim \mathcal{N}(0,1)$, often results in “disentangled” representations.

This effect can be reinforced with a greater weight of the KL term

$$\mathcal{L} = \beta \mathbb{E}_{q(X)} \left[ \text{KL} (q(Z | X) \| p(Z)) \right] - \mathbb{E}_{q(X,Z)} \left[ \log p(X | Z) \right],$$

resulting in the $\beta$-VAE proposed by Higgins et al. (2017).
We propose augmenting the original VAE framework with a single hyperparameter \( \beta \) with permission. (b). Only \( \beta = 1 \) achieves state of the art disentangling performance against both the best unsupervised (InfoGAN: Goodfellow et al., 2014) framework to additionally maximise the mutual information between a categorical latent variables. Only \( \beta = 1 \) also lacks a principled way of measuring the degree of disentanglement achieved by different models or when optimising the hyperparameters of a single model.

Our main contributions are the following: 1) we propose \( \text{\( \beta \)-VAE} \) approach achieves state-of-the-art disentanglement performance compared to various baselines on a variety of complex datasets. 2) we devise a new measure of disentanglement and show that \( \text{\( \beta \)-VAE} \) learns an entangled representation (e.g. chair width is entangled with azimuth and leg style (b)). InfoGAN traversal is over ten dimensional \([-1, 1]\) range. VAE always performs worse than \( \text{\( \beta \)-VAE} \) (Higgins et al., 2017), prompting the development of more elaborate semi-supervised VAE-based approaches for disentangled representation of these factors. The reliance of InfoGAN on the GAN framework, however, comes with several drawbacks: 1) it only learns disentangled representations of independent visual data generative factors; 2) we devise a measure of disentanglement in learnt latent representations compared to the unmodified VAE framework (Kingma & Welling, 2014; Rezende et al., 2014). Furthermore, we show that \( \text{\( \beta \)-VAE} \) framework (Kingma & Welling, 2014; Rezende et al., 2014) and faces (Paysan et al., 2009) using qualitative evaluation. Finally, to help quantify the differences, we develop a new measure of disentanglement and show that \( \text{\( \beta \)-VAE} \) outperforms all our baselines on this measure (ICA, PCA, VAE Kingma & Ba (2014), DC-IGN Kulkarni et al., 2015), and InfoGAN (Chen et al., 2016)).

\( \text{\( \beta \)-VAE} \) learns the correct disentangled representation (c) leg style. InfoGAN traversal is over ten dimensional \([-1, 1]\) range. VAE always performs worse than \( \text{\( \beta \)-VAE} \) (Higgins et al., 2017).
References
