CS598LAZ - Variational Autoencoders
Raymond Yeh, Junting Lou, Teck-Yian Lim
Outline

- Review Generative Adversarial Network
- Introduce Variational Autoencoder (VAE)
- VAE applications
  - VAE + GANs
- Introduce Conditional VAE (CVAE)
- Conditional VAE applications.
  - Attribute2Image
  - Diverse Colorization
  - Forecasting motion
- Take aways
Last lecture we discussed **generative models**

- **Task:** Given a dataset of images \( \{X_1, X_2, \ldots\} \) can we learn the distribution of \( X \)?
- Typically generative models implies modelling \( P(X) \).
  - Very limited, given an image the model outputs a probability
- More Interested in models which we can **sample** from.
  - Can generate random examples that follow the distribution of \( P(X) \).
Recap: Generative Model + GAN

Recap: Generative Adversarial Network

- **Pro**: Do not have to explicitly specify a form on $P(X|z)$, $z$ is the latent space.
- **Con**: Given a desired image, difficult to map back to the latent variable.

Image Credit: Last lecture
Manifold Hypothesis

Natural data (high dimensional) actually lies in a low dimensional space.
Variational Autoencoder (VAE)

Variational Autoencoder (2013) work prior to GANs (2014)

- Explicit Modelling of $P(X|z; \theta)$, we will drop the $\theta$ in the notation.
- $z \sim P(z)$, which we can sample from, such as a Gaussian distribution.

$$P(X) = \int P(X|z; \theta)P(z)dz$$

- Maximum Likelihood --- Find $\theta$ to maximize $P(X)$, where $X$ is the data.
- Approximate with samples of $z$

$$P(X) \approx \frac{1}{n} \sum_{i=0}^{n} P(X|z_i)$$
Variational Autoencoder (VAE)

Variational Autoencoder (2013) work prior to GANs (2014)

- **Explicit Modelling of** $P(X|z; \theta)$, we will drop the $\theta$ in the notation.
- $z \sim P(z)$, which we can sample from, such as a Gaussian distribution.

\[
P(X) = \int P(X|z; \theta)P(z)dz
\]

- Maximum Likelihood --- **Find $\theta$ to maximize** $P(X)$, where $X$ is the data.
- **Approximate with samples of** $z$

\[
P(X) \approx \frac{1}{n} \sum_{i=0}^{n} P(X|z_i)
\]
Variational Autoencoder (VAE)

- Approximate with samples of $z$

$$P(X) \approx \frac{1}{n} \sum_{i=0}^{n} P(X|z_i)$$

- Need a lot of samples of $z$ and most of the $P(X|z) \approx 0$.
- Not practical computationally.
- **Question**: Is it possible to know which $z$ will generate $P(X|z) >> 0$?
  - Learn a distribution $Q(z)$, where $z \sim Q(z)$ generates $P(X|z) >> 0$. 
Variational Autoencoder (VAE)

- Approximate with samples of $z$

\[ P(X) \approx \frac{1}{n} \sum_{i=0}^{n} P(X|z_i) \]

- Need a lot of samples of $z$ and most of the $P(X|z) \approx 0$.
- Not practical computationally.
- **Question:** Is it possible to know which $z$ will generate $P(X|z) >> 0$?
  - Learn a distribution $Q(z)$, where $z \sim Q(z)$ generates $P(X|z) >> 0$.
Variational Autoencoder (VAE)

- Approximate with samples of $z$

$$P(X) \approx \frac{1}{n} \sum_{i=0}^{n} P(X|z_i)$$

- Need a lot of samples of $z$ and most of the $P(X|z) \approx 0$.
- Not practical computationally.

- **Question:** Is it possible to know which $z$ will generate $P(X|z) >> 0$?
  - Learn a distribution $Q(z)$, where $z \sim Q(z)$ generates $P(X|z) >> 0$. 
Variational Autoencoder (VAE)

Assume we can learn a distribution $Q(z)$, where $z \sim Q(z)$ generates $P(X|z) \gg 0$

- We want $P(X) = E_{z \sim P(z)} P(X|z)$, but not practical. 

- We can compute $E_{z \sim Q(z)} P(X|z)$, more practical.

- **Question:** How does $E_{z \sim Q(z)} P(X|z)$ and $P(X)$ relate?

  - In the following slides, we derive the following relationship

\[
\log P(X) - \mathcal{D} [Q(z) \| P(z|X)] = E_{z \sim Q} \left[ \log P(X|z) \right] - \mathcal{D} [Q(z) \| P(z)]
\]
Variational Autoencoder (VAE)

Assume we can learn a distribution $Q(z)$, where $z \sim Q(z)$ generates $P(X|z) >> 0$

- We want $P(X) = \mathbb{E}_{z \sim P(z)} P(X|z)$, but not practical.

- We can compute $\mathbb{E}_{z \sim Q(z)} P(X|z)$, more practical.

- **Question:** How does $\mathbb{E}_{z \sim Q(z)} P(X|z)$ and $P(X)$ relate?

  - In the following slides, we derive the following relationship

\[
\log P(X) - \mathcal{D} [Q(z) \| P(z|X)] = \mathbb{E}_{z \sim Q} [\log P(X|z)] - \mathcal{D} [Q(z) \| P(z)]
\]
Variational Autoencoder (VAE)

Assume we can learn a distribution $Q(z)$, where $z \sim Q(z)$ generates $P(X|z) \gg 0$

- We want $P(X) = \mathbb{E}_{z \sim P(z)}P(X|z)$, but not practical.

- We can compute $\mathbb{E}_{z \sim Q(z)}P(X|z)$, more practical.

- **Question:** How does $\mathbb{E}_{z \sim Q(z)}P(X|z)$ and $P(X)$ relate?

  - In the following slides, we derive the following relationship

$$\log P(X) - \mathcal{D}[Q(z) \| P(z|X)] = \mathbb{E}_{z \sim Q} [\log P(X|z)] - \mathcal{D}[Q(z) \| P(z)]$$
Variational Autoencoder (VAE)

**We want** $P(X) = E_{z \sim P(z)} P(X|z)$, but not practical.

**We can** compute $E_{z \sim Q(z)} P(X|z)$, more practical.

**Question**: How does $E_{z \sim Q(z)} P(X|z)$ and $P(X)$ relate?

- In the following slides, we derive the following relationship

$$
\log P(X) - \mathcal{D} [Q(z) \| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - \mathcal{D} [Q(z) \| P(z)]
$$
Variational Autoencoder (VAE)

Assume we can learn a distribution \( Q(z) \), where \( z \sim Q(z) \) generates \( P(X|z) \gg 0 \)

- We want \( P(X) = E_{z \sim P(z)} P(X|z) \), but not practical. \[ P(X) \approx \frac{1}{n} \sum_{i=0}^{n} P(X|z_i) \]

- We \textbf{can} compute \( E_{z \sim Q(z)} P(X|z) \), more practical.

- \textbf{Question}: How does \( E_{z \sim Q(z)} P(X|z) \) and \( P(X) \) relate?

  - In the following slides, we derive the following relationship

\[
\log P(X) - D [Q(z) || P(z|X)] = E_{z \sim Q} [\log P(X|z)] - D [Q(z) || P(z)]
\]
Definition of KL divergence:

\[ D[Q(z) \parallel P(z|X)] = E_{z \sim Q} [\log Q(z) - \log P(z|X)] \]

- Apply Bayes Rule on \( P(z|X) \) and substitute into the equation above.
  - \( P(z|X) = P(X|z) P(z) / P(X) \)
  - \( \log (P(z|X)) = \log P(X|z) + \log P(z) - \log P(X) \)
  - \( P(X) \) does not depend on \( z \), can take it outside of \( E_{z \sim Q} \)

\[ D[Q(z) \parallel P(z|X)] = E_{z \sim Q} [\log Q(z) - \log P(X|z) - \log P(z)] + \log P(X) \]
Relating $E_{z \sim Q(z)} P(X|z)$ and $P(X)$

- Definition of KL divergence:

$$
\mathcal{D} [Q(z) \parallel P(z|X)] = E_{z \sim Q} \left[ \log Q(z) - \log P(z|X) \right]
$$

- Apply **Bayes Rule** on $P(z|X)$ and substitute into the equation above.
  - $P(z|X) = P(X|z) P(z) / P(X)$
  - $\log (P(z|X)) = \log P(X|z) + \log P(z) - \log P(X)$
  - $P(X)$ does not depend on $z$, can take it outside of $E_{z \sim Q}$

$$
\mathcal{D} [Q(z) \parallel P(z|X)] = E_{z \sim Q} \left[ \log Q(z) - \log P(X|z) - \log P(z) \right] + \log P(X)
$$
- Definition of KL divergence:

\[ \mathcal{D} [Q(z) \| P(z|X)] = E_{z \sim Q} [\log Q(z) - \log P(z|X)] \]

- Apply **Bayes Rule** on \( P(z|X) \) and substitute into the equation above.
  - \( P(z|X) = P(X|z) P(z) / P(X) \)
  - \( \log (P(z|X)) = \log P(X|z) + \log P(z) - \log P(X) \)
  - \( P(X) \) does not depend on \( z \), can take it outside of \( E_{z \sim Q} \)

\[ \mathcal{D} [Q(z) \| P(z|X)] = E_{z \sim Q} [\log Q(z) - \log P(X|z) - \log P(z)] + \log P(X) \]
Relating $E_{z \sim Q(z)} P(X|z)$ and $P(X)$

- Definition of KL divergence:
  \[
  D [Q(z) \| P(z|X)] = E_{z \sim Q} \left[ \log Q(z) - \log P(z|X) \right]
  \]

- Apply **Bayes Rule on** $P(z|X)$ and substitute into the equation above.
  - $P(z|X) = P(X|z) P(z) / P(X)$
  - $\log (P(z|X)) = \log P(X|z) + \log P(z) - \log P(X)$
  - $P(X)$ does not depend on $z$, can take it outside of $E_{z \sim Q}$

\[
D [Q(z) \| P(z|X)] = E_{z \sim Q} \left[ \log Q(z) - \log P(X|z) - \log P(z) \right] + \log P(X)
\]
Relating $E_{z \sim Q(z)} P(X|z)$ and $P(X)$

- Definition of KL divergence:
  \[
  \mathcal{D} [Q(z) \parallel P(z|X)] = \mathbb{E}_{z \sim Q} [\log Q(z) - \log P(z|X)]
  \]

- Apply **Bayes Rule on** $P(z|X)$ and substitute into the equation above.
  - $P(z|X) = P(X|z) P(z) / P(X)$
  - $\log (P(z|X)) = \log P(X|z) + \log P(z) - \log P(X)$
  - $P(X)$ does not depend on $z$, can take it outside of $E_{z \sim Q}$

\[
\mathcal{D} [Q(z) \parallel P(z|X)] = \mathbb{E}_{z \sim Q} [\log Q(z) - \log P(X|z) - \log P(z)] + \log P(X)
\]
Relating $E_{z \sim Q(z)} P(X|z)$ and $P(X)$

$$\mathcal{D} [Q(z) \| P(z|X)] = E_{z \sim Q} [\log Q(z) - \log P(X|z) - \log P(z)] + \log P(X)$$

Rearrange the terms:

$$E_{z \sim Q} [\log Q(z) - \log P(z)] = \mathcal{D} [Q(z) \| P(z)]$$

$$\log P(X) - \mathcal{D} [Q(z) \| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - \mathcal{D} [Q(z) \| P(z)]$$
Rearrange the terms:

$$D \left[ Q(z) \mid \mid P(z \mid X) \right] = E_{z \sim Q} \left[ \log Q(z) - \log P(X \mid z) - \log P(z) \right] + \log P(X)$$

Relating $E_{z \sim Q(z)} P(X \mid z)$ and $P(X)$

Rearrange the terms:

$$E_{z \sim Q} \left[ \log Q(z) - \log P(z) \right] = D \left[ Q(z) \mid \mid P(z) \right]$$

$$\log P(X) - D \left[ Q(z) \mid \mid P(z \mid X) \right] = E_{z \sim Q} \left[ \log P(X \mid z) \right] - D \left[ Q(z) \mid \mid P(z) \right]$$
Relating $E_{z \sim Q(z)} P(X|z)$ and $P(X)$

$$D[Q(z)\|P(z|X)] = E_{z \sim Q} [\log Q(z) - \log P(X|z) - \log P(z)] + \log P(X)$$

Rearrange the terms:

$$E_{z \sim Q} [\log Q(z) - \log P(z)] = D[Q(z)\|P(z)]$$

$$\log P(X) - D[Q(z)\|P(z|X)] = E_{z \sim Q} [\log P(X|z)] - D[Q(z)\|P(z)]$$
Relating $E_{z \sim Q(z)} P(X|z)$ and $P(X)$

$$
\mathcal{D} [Q(z) \| P(z|X)] = E_{z \sim Q} [\log Q(z) - \log P(X|z) - \log P(z)] + \log P(X)
$$

Rearrange the terms:

$$
E_{z \sim Q} [\log Q(z) - \log P(z)] = \mathcal{D} [Q(z) \| P(z)]
$$

$$
\log P(X) - \mathcal{D} [Q(z) \| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - \mathcal{D} [Q(z) \| P(z)]
$$
Relating \( E_{z \sim Q(z)} P(X|z) \) and \( P(X) \)

\[
\mathcal{D} [Q(z) || P(z|X)] = E_{z \sim Q} [\log Q(z) - \log P(X|z) - \log P(z)] + \log P(X)
\]

Rearrange the terms:

\[
E_{z \sim Q} [\log Q(z) - \log P(z)] = \mathcal{D} [Q(z) || P(z)]
\]

\[
\log P(X) - \mathcal{D} [Q(z) || P(z|X)] = E_{z \sim Q} [\log P(X|z)] - \mathcal{D} [Q(z) || P(z)]
\]
Intuition

Why is this important?

\[
\log P(X) - D[Q(z) \| P(z | X)] = E_{z \sim Q} [\log P(X | z)] - D[Q(z) \| P(z)]
\]

- Recall we want to maximize \( P(X) \) with respect to \( \theta \), which we cannot do.
- KL divergence is always \( > 0 \).
- \( \log P(X) > \log P(X) - D[Q(z) \| P(z | X)] \).
- Maximize the lower bound instead.
- **Question:** How do we get \( Q(z) \) ?
Intuition

Why is this important?

\[
\log P(X) - \mathcal{D} [Q(z) \| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - \mathcal{D} [Q(z) \| P(z)]
\]

- Recall we want to maximize \( P(X) \) with respect to \( \theta \), which we cannot do.
- KL divergence is always \( > 0 \).
- \( \log P(X) > \log P(X) - \mathcal{D}[Q(z) \| P(z|X)] \).
- Maximize the lower bound instead.
- **Question:** How do we get \( Q(z) \)?
Why is this important?

\[
\log P(X) - \mathcal{D} [Q(z) \| P(z \| X)] = E_{z \sim Q} [\log P(X \| z)] - \mathcal{D} [Q(z) \| P(z)]
\]

- Recall we want to maximize \( P(X) \) with respect to \( \theta \), which we cannot do.
- KL divergence is always \( > 0 \).
- \( \log P(X) > \log P(X) - \mathcal{D}[Q(z) \| P(z \| X)] \).
- Maximize the lower bound instead.
- **Question:** How do we get \( Q(z) \)?

**Intuition**
Why is this important?

- Recall we want to **maximize** \( P(X) \) with respect to \( \theta \), **which we cannot do**.
- KL divergence is always > 0.
- \( \log P(X) > \log P(X) - D[Q(z) \parallel P(z|X)] \).
- Maximize the **lower bound** instead.
- **Question:** How do we get \( Q(z) \)?
Why is this important?

Recall we want to maximize $P(X)$ with respect to $\theta$, which we cannot do.

KL divergence is always $> 0$.

$log P(X) > log P(X) - D[Q(z) \parallel P(z|X)]$.

Maximize the lower bound instead.

Question: How do we get $Q(z)$?
How to Get Q(z)?

\[
\log P(X) - D [Q(z) \| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - D [Q(z) \| P(z)]
\]

- Q(z) or Q(z|X)?
- Model Q(z|X) with a neural network.
- Assume Q(z|X) to be Gaussian, N(\mu, c \cdot I)
  - Neural network outputs the mean \( \mu \) and diagonal covariance matrix \( c \cdot I \).
  - **Input:** Image, **Output:** Distribution

Let’s call Q(z|X) the **Encoder**.
How to Get Q(z)?

- Q(z) or Q(z|X)?
- Model Q(z|X) with a neural network.
- Assume Q(z|X) to be Gaussian, \( N(\mu, c \cdot I) \)
  - Neural network outputs the mean \( \mu \), and diagonal covariance matrix \( c \cdot I \).
  - Input: Image, Output: Distribution

Let's call Q(z|X) the Encoder.
How to Get Q(z)?

\[ \log P(X) - D[Q(z) \| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - D[Q(z) \| P(z)] \]

- Q(z) or Q(z|X)?
- Model Q(z|X) with a neural network.
- Assume Q(z|X) to be Gaussian, \( N(\mu, c \cdot I) \)
  - Neural network outputs the mean \( \mu \), and diagonal covariance matrix \( c \cdot I \).
  - Input: Image, Output: Distribution

Let's call Q(z|X) the Encoder.
How to Get $Q(z)$?

- $Q(z)$ or $Q(z|X)$?
- Model $Q(z|X)$ with a neural network.
- Assume $Q(z|X)$ to be Gaussian, $N(\mu, c \cdot I)$
  - Neural network outputs the mean $\mu$, and diagonal covariance matrix $c \cdot I$.
  - Input: Image, Output: Distribution

Let’s call $Q(z|X)$ the Encoder.
How to Get Q(z)?

- Q(z) or Q(z|X)?
- Model Q(z|X) with a neural network.
- Assume Q(z|X) to be Gaussian, $N(\mu, c \cdot I)$
  - Neural network outputs the mean $\mu$, and diagonal covariance matrix $c \cdot I$.
  - Input: Image, Output: Distribution

Let’s call Q(z|X) the Encoder.
How to Get $Q(z)$?

- $Q(z)$ or $Q(z|X)$?

- Model $Q(z|X)$ with a neural network.

- Assume $Q(z|X)$ to be Gaussian, $N(\mu, c \cdot I)$
  - Neural network outputs the mean $\mu$, and diagonal covariance matrix $c \cdot I$.
  - Input: Image, Output: Distribution

Let’s call $Q(z|X)$ the Encoder.
Convert the lower bound to a loss function:

$$\log P(X) - D[Q(z)\|P(z|X)] = E_{z \sim Q} [\log P(X|z)] - D[Q(z)\|P(z)]$$

- Model $P(X|z)$ with a neural network, let $f(z)$ be the network output.
- Assume $P(X|z)$ to be i.i.d. Gaussian
  - $X = f(z) + \eta$, where $\eta \sim N(0,I)$ *Think Linear Regression*
  - Simplifies to an $l_2$ loss: $||X-f(z)||^2$

Let's call $P(X|z)$ the Decoder.
Convert the lower bound to a loss function:

\[
\log P(X) - D [Q(z)\|P(z|X)] = E_{z \sim Q} [\log P(X|z)] - D [Q(z)\|P(z)]
\]

- Model \( P(X|z) \) with a neural network, let \( f(z) \) be the network output.
- Assume \( P(X|z) \) to be i.i.d. Gaussian
  - \( X = f(z) + \eta \), where \( \eta \sim N(0,I) \) *Think Linear Regression*
  - Simplifies to an \( l_2 \) loss: \( ||X-f(z)||^2 \)

Let's call \( P(X|z) \) the Decoder.
Convert the lower bound to a loss function:

\[
\log P(X) - \mathcal{D}[Q(z) \| P(z|X)] = E_{z \sim Q}[\log P(X|z)] - \mathcal{D}[Q(z) \| P(z)]
\]

- Model \( P(X|z) \) with a neural network, let \( f(z) \) be the network output.
- Assume \( P(X|z) \) to be i.i.d. **Gaussian**
  - \( X = f(z) + \eta \), where \( \eta \sim N(0,I) \)    *Think Linear Regression*
  - Simplifies to an \( l_2 \) loss: \( \|X-f(z)\|^2 \)

Let’s call \( P(X|z) \) the Decoder.
Convert the lower bound to a loss function:

$$\log P(X) - D [Q(z)\| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - D [Q(z)\| P(z)]$$

- Model $P(X|z)$ with a neural network, let $f(z)$ be the network output.
- Assume $P(X|z)$ to be i.i.d. **Gaussian**
  - $X = f(z) + \eta$, where $\eta \sim N(0,I)$  *Think Linear Regression*
  - Simplifies to an $l_2$ loss: $||X-f(z)||^2$

Let’s call $P(X|z)$ the Decoder.
Convert the lower bound to a loss function:

\[
\log P(X) - D [Q(z) \| P(z \| X)] = E_{z \sim Q} [\log P(X \| z)] - D [Q(z) \| P(z)]
\]

- Model \( P(X|z) \) with a neural network, let \( f(z) \) be the network output.
- Assume \( P(X|z) \) to be i.i.d. \textbf{Gaussian}
  - \( X = f(z) + \eta \), where \( \eta \sim N(0, I) \)  \hspace{1em} *Think Linear Regression*
  - \textbf{Simplifies to an} \( I_2 \) \textbf{loss:} \( ||X - f(z)||^2 \)

Let’s call \( P(X|z) \) the Decoder.
Convert the lower bound to a loss function:

$$\log P(X) - D [Q(z) \| P(z|X)] = E_{z \sim Q} [\log P(X|z)] - D [Q(z) \| P(z)]$$

- Model $P(X|z)$ with a neural network, let $f(z)$ be the network output.
- Assume $P(X|z)$ to be i.i.d. **Gaussian**
  - $X = f(z) + \eta$, where $\eta \sim N(0, I)$  *Think Linear Regression*
  - **Simplifies to an $l_2$ loss:** $||X - f(z)||^2$

Let’s call $P(X|z)$ the Decoder.
VAE’s Loss function

Convert the lower bound to a loss function:
\[
\log P(X) - \mathcal{D}[Q(z) \| P(z|X)] = E_{z \sim Q} \left[ \log P(X|z) \right] - \mathcal{D}[Q(z) \| P(z)]
\]

Assume \( P(z) \sim N(0,I) \) then \( \mathcal{D}[Q(z|X) \| P(z)] \) has a closed form solution.

Putting it all together:
\[
L = ||X - f(z)||^2 - \lambda \cdot \mathcal{D}[Q(z) \| P(z)]
\]

, given a \((X, z)\) pair.

Pixel difference

Regularization
VAE’s Loss function

Convert the lower bound to a loss function:

$$\log P(X) - D[Q(z)||P(z|X)] = E_{z \sim Q}[\log P(X|z)] - D[Q(z)||P(z)]$$

Assume $P(z) \sim N(0, I)$ then $D[Q(z|X) \parallel P(z)]$ has a closed form solution.

Putting it all together:

$$E_{z \sim Q(z|X)} \log P(X|z) \propto ||X-f(z)||^2$$

$$L = ||X - f(z)||^2 - \lambda \cdot D[Q(z) \parallel P(z)]$$

, given a $(X, z)$ pair.

Pixel difference

Regularization
VAE’s Loss function

Convert the lower bound to a loss function:

\[
\log P(X) - D[Q(z) || P(z|X)] = E_{z \sim Q} [\log P(X|z)] - D[Q(z) || P(z)]
\]

Assume \( P(z) \sim N(0,I) \) then \( D[Q(z|X) || P(z)] \) has a closed form solution.

Putting it all together:

\[
E_{z \sim Q(z|X)} \log P(X|z) \propto \|X - f(z)\|^2
\]

\[
L = \|X - f(z)\|^2 - \lambda \cdot D[Q(z) || P(z)]
\]

, given a \((X, z)\) pair.
VAE's Loss function

Convert the lower bound to a loss function:

\[
\log P(X) - \mathcal{D}[Q(z) \| P(z|X)] = E_{z \sim Q}[\log P(X|z)] - \mathcal{D}[Q(z) \| P(z)]
\]

Assume \( P(z) \sim \mathcal{N}(0,I) \) then \( \mathcal{D}[Q(z|X) \| P(z)] \) has a closed form solution.

Putting it all together:

\[
\mathbb{E}_{z \sim Q(z|X)} \log P(X|z) \propto \|X - f(z)\|^2
\]

\[
L = \|X - f(z)\|^2 - \lambda \cdot \mathcal{D}[Q(z) \| P(z)]
\]

, given a \((X, z)\) pair.
Training the **Decoder** is easy, just standard backpropagation.

How to train the **Encoder**?
- Not obvious how to apply gradient descent through samples.
Reparameterization Trick

How to effectively backpropagate through the z samples to the Encoder?

Reparametrization Trick

- \( z \sim N(\mu, \sigma) \) is equivalent to
- \( \mu + \sigma \cdot \varepsilon \), where \( \varepsilon \sim N(0, 1) \)
- Now we can easily backpropagate the loss to the Encoder.

Image Credit: Tutorial on VAEs
VAE Training

Given a dataset of examples $X = \{X_1, X_2...\}$

Initialize parameters for Encoder and Decoder

Repeat till convergence:

1. $X^M$ <-- Random minibatch of $M$ examples from $X$
2. $\epsilon$ <-- Sample $M$ noise vectors from $N(0, I)$
3. Compute $L(X^M, \epsilon, \theta)$ (i.e. run a forward pass in the neural network)
4. Gradient descent on $L$ to updated Encoder and Decoder.
Given a dataset of examples $\mathbf{X} = \{X_1, X_2\ldots\}$

Initialize parameters for Encoder and Decoder

Repeat till convergence:

$\mathbf{X}^M \leftarrow$ Random minibatch of $M$ examples from $\mathbf{X}$

$\mathbf{\epsilon} \leftarrow$ Sample $M$ noise vectors from $N(0, I)$

Compute $L(\mathbf{X}^M, \mathbf{\epsilon}, \theta)$ (i.e. run a forward pass in the neural network)

Gradient descent on $L$ to updated Encoder and Decoder.
Given a dataset of examples $X = \{X_1, X_2\ldots\}$

Initialize parameters for Encoder and Decoder

Repeat till convergence:

$X^M \leftarrow$ Random minibatch of $M$ examples from $X$

$\epsilon \leftarrow$ Sample $M$ noise vectors from $N(0, I)$

Compute $L(X^M, \epsilon, \theta)$ (i.e. run a forward pass in the neural network)

Gradient descent on $L$ to update Encoder and Decoder.
Given a dataset of examples $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2\ldots\}$

Initialize parameters for Encoder and Decoder

Repeat till convergence:

$\mathbf{X}^M \leftarrow$ Random minibatch of M examples from $\mathbf{X}$

$\mathbf{\epsilon} \leftarrow$ Sample M noise vectors from $\mathcal{N}(0, \mathbf{I})$

Compute $L(\mathbf{X}^M, \mathbf{\epsilon}, \theta)$ (i.e. run a forward pass in the neural network)

Gradient descent on $L$ to updated Encoder and Decoder.
Given a dataset of examples $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2\ldots\}$

Initialize parameters for Encoder and Decoder

Repeat till convergence:

1. $\mathbf{X}^M \leftarrow$ Random minibatch of $M$ examples from $\mathbf{X}$
2. $\epsilon \leftarrow$ Sample $M$ noise vectors from $\mathcal{N}(0, \mathbf{I})$
3. Compute $L(\mathbf{X}^M, \epsilon, \theta)$ (i.e. run a forward pass in the neural network)
4. Gradient descent on $L$ to updated Encoder and Decoder.
At test-time, we want to evaluate the performance of VAE to generate a new sample. Remove the Encoder, as no test-image for generation task. Sample $z \sim N(0,I)$ and pass it through the Decoder. No good quantitative metric, relies on visual inspection.
At test-time, we want to evaluate the **performance of VAE to generate a new sample**.

- Remove the Encoder, as no test-image for generation task.
- Sample $z \sim N(0,I)$ and pass it through the Decoder.
- No good quantitative metric, relies on visual inspection.
- At test-time, we want to evaluate the **performance of VAE to generate a new sample**.
- Remove the Encoder, as **no test-image** for generation task.
- Sample $z \sim N(0,I)$ and pass it through the Decoder.
- No good quantitative metric, relies on visual inspection.
- At test-time, we want to evaluate the **performance of VAE to generate a new sample.**
- Remove the Encoder, as **no test-image** for generation task.
- Sample $z \sim \mathcal{N}(0,I)$ and pass it through the Decoder.
- No good quantitative metric, relies on visual inspection.
- At test-time, we want to evaluate the performance of VAE to generate a new sample.
- Remove the Encoder, as no test-image for generation task.
- Sample $z \sim N(0, I)$ and pass it through the Decoder.
- No good quantitative metric, relies on visual inspection.
Common VAE architecture

Fully Connected (Initially Proposed)

Common Architecture (convolutional) similar to DCGAN.
Disentangle latent factor

Autoencoder can disentangle latent factors [MNIST DEMO]:

[Image Credit: Auto-encoding Variational Bayes]
Disentangle latent factor

Image Credit: Deep Convolutional Inverse Graphics Network
We have seen **very similar results** during last lecture: InfoGan.
VAE vs. GAN

Image Credit: Autoencoding beyond pixels using a learned similarity metric
**VAE vs. GAN**

**VAE**
- ✓: Given an X easy to find $z$.
- ✓: Interpretable probability $P(X)$
- X: Usually outputs blurry Images

**GAN**
- ✓: Very sharp images
- X: Given an X difficult to find $z$. (Need to backprop.)
- ✓/X: No explicit $P(X)$.

![Image Credit: Autoencoding beyond pixels using a learned similarity metric](image_url)
GAN + VAE (Best of both models)

Encoder → z → Decoder / Generator → Discriminator

KL Divergence

L2 Difference

$\mathcal{L} = \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{kl like}} + \mathcal{L}_{\text{GAN}}$

Image Credit: Autoencoding beyond pixels using a learned similarity metric
VAE Disl: Train a GAN first, then use the discriminator of GAN to train a VAE.

VAE/GAN: GAN and VAE trained together.
Conditional VAE (CVAE)

What if we have labels? (e.g. digit labels or attributes) Or other inputs we wish to condition on (Y).

- None of the derivation changes.
- Replace all $P(X|z)$ with $P(X|z,Y)$.
- Replace all $Q(z|X)$ with $Q(z|X,Y)$.
- Go through the same KL divergence procedure, to get the same lower bound.
What if we have **labels**? (e.g. digit labels or attributes) Or other inputs we wish to condition on \((Y)\).

- None of the derivation changes.
- Replace all \(P(X|z)\) with \(P(X|z,Y)\).
- Replace all \(Q(z|X)\) with \(Q(z|X,Y)\).
- Go through the same KL divergence procedure, to get the same lower bound.
What if we have **labels**? (e.g. digit labels or attributes) Or other inputs we wish to condition on \((Y)\).

- **NONE** of the derivation changes.
- Replace all \(P(X|z)\) with \(P(X|z,Y)\).
- Replace all \(Q(z|X)\) with \(Q(z|X,Y)\).
- Go through the same KL divergence procedure, to get the same lower bound.
Common CVAE architecture

Common Architecture (convolutional) for CVAE
- Again, remove the **Encoder** as test time
- **Sample** $z \sim N(0, I)$ and **input** a **desired** $Y$ to the **Decoder**.
a young girl with brown hair is smiling.
Attribute-conditioned image progression

(a) progression on gender
(b) progression on age
(c) progression on expression
(d) progression on eyewear
(e) progression on hair color
(f) progression on primary color

$p_\theta(x|y, z)$ with $z \sim \mathcal{N}(0, I)$ and $y = [y_\alpha, y_{\text{rest}}]$, where $y_\alpha = (1-\alpha) \cdot y_{\text{min}} + \alpha \cdot y_{\text{max}}$
Image Colorization

- An ambiguous problem

Image Colorization

- An ambiguous problem

Blue? Red? Yellow?
Goal:

Learn a conditional model $P(C|G)$

Color field $C$, given grey level image $G$

Next, draw samples from $\{C_k\}_{k=1}^N \sim P(C|G)$ to obtain diverse colorization
Goal:

Learn a conditional model $P(C|G)$

Color field $C$, given grey level image $G$

Next, draw samples from $\{C_k\}_{k=1}^N \sim P(C|G)$ to obtain diverse colorization

Difficult to learn!

Exceedingly high dimensions! (Curse of dimensionality)
Goal:

Learn a conditional model $P(C|G)$

Color field $C$, given grey level image $G$.

Instead of learning $C$ directly, learn a low-dimensional embedding variable $z$ (VAE).

Using another network, learn $P(z|G)$.
  - Use a Mixture Density Network (MDN)
    - Good for learning multi-modal conditional model.

At test time, use VAE decoder to obtain $C_k$ for each $z_k$. 
Architecture

Training Procedure

Step 1

Encoder  \rightarrow z \rightarrow \text{Decoder}

Step 2

MDN  \rightarrow \text{Color Image (C)}

Testing Procedure

Sampling

\[ z_1, z_2, z_3 \]

Decoder

\[ C_1, C_2, C_3 \]

Multiple Viable Colorizations

Gray Image (G)

Image Credit: Learning Diverse Image Colorization
**Step 1:** Learn a low dimensional $z$ for color.
- Standard VAE: Overly smooth and "washed out", as training using $L_2$ loss directly on the color space.

Authors introduced several new loss functions to solve this problem.

1. Weighted $L_2$ on the color space to encourage "color" diversity. Weighting the very common color smaller.
2. Top-$k$ principal components, $P_k$, of the color space. Minimize the $L_2$ of the projection.
3. Encourage color fields with the same gradient as ground truth.

\[
\mathcal{L}_{dec} = \mathcal{L}_{hist} + \lambda_{mah} \mathcal{L}_{mah} + \lambda_{grad} \mathcal{L}_{grad}
\]
Step 1: Learn a low dimensional $z$ for color.
- Standard VAE: Overly smooth and “washed out”, as training using $L_2$ loss directly on the color space.

Authors introduced several new loss functions to solve this problem.
1. Weighted $L_2$ on the color space to encourage “color” diversity. Weighting the very common color smaller.
2. Top-k principal components, $P_k$, of the color space. Minimize the $L_2$ of the projection.
3. Encourage color fields with the same gradient as ground truth.

$$L_{dec} = [L_{hist}] + \lambda_{mah}L_{mah} + \lambda_{grad}L_{grad}$$
Devil is in the details

**Step 1**: Learn a low dimensional \( z \) for color.
- Standard VAE: Overly smooth and “washed out”, as training using \( L_2 \) loss directly on the color space.

Authors introduced several new loss functions to solve this problem.

1. **Weighted \( L_2 \)** on the color space to encourage “color” diversity. Weighting the very common color smaller.
2. **Top-k principal components**, \( P_k \), of the color space. Minimize the \( L_2 \) of the projection.
3. Encourage color fields with the same gradient as ground truth.

\[
\mathcal{L}_{dec} = \mathcal{L}_{hist} + \lambda_{mah}\mathcal{L}_{mah} + \lambda_{grad}\mathcal{L}_{grad}
\]
**Devil is in the details**

**Step 1:** Learn a low dimensional $z$ for color.
- Standard VAE: Overly smooth and “washed out”, as training using $L_2$ loss directly on the color space.

Authors introduced several new loss functions to solve this problem.

1. **Weighted $L_2$** on the color space to encourage “color” diversity. Weighting the very common color smaller.
2. **Top-k principal components, $P_k$**, of the color space. Minimize the $L_2$ of the projection.
3. **Encourage color fields with the same gradient as ground truth.**

$$L_{dec} = L_{hist} + \lambda_{mah} L_{mah} + \lambda_{grad} L_{grad}$$
Step 2: Conditional Model: Grey-level to Embedding

\[ \mathcal{L}_{mdn} = -\log P(z|G) = -\log \sum_{i=1}^{M} \pi_i(G, \phi) \mathcal{N}(z|\mu_i(G, \phi), \sigma) \]

- Learn a multimodal distribution
- At test time sample at each mode to generate diversity.
- Similar to CVAE, but this has more “explicit” modeling of the P(z|G).
- Comparison with CVAE, condition on the gray scale image.
Effects of Loss Terms

$L_2$ Loss

$L_{man}$

All Terms (Equation 4)

Ground Truth

LFW Dataset | LSUN Church Dataset | Imagenet-Val Dataset

Image Credit: Learning Diverse Image Colorization
- Given an image, humans can often infer how the objects in the image might move
- Modeled as dense trajectories of how each pixel will move over time
- Given an image, humans can often infer how the objects in the image might move
- Modeled as dense trajectories of how each pixel will move over time

Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs
Applications: Forecasting from Static Images

Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs
Applications: Forecasting from Static Images

Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs
Forecasting from Static Images

- Given an image, humans can often infer how the objects in the image might move.
- Modeled as dense trajectories of how each pixel will move over time.
- Why is this difficult?
  - Multiple possible solutions
- Recall that latent space can encode information not in the image
  - By using CVAEs, multiple possibilities can be generated
Forecasting from Static Images

(a) Trajectories on Image
(b) Trajectories in Space-Time

Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs
Architecture

(a) Testing Architecture

(b) Training Architecture

Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs
Computed Optical Flow

Parameters From Image

Learnt distributions of trajectories

Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs
Image Tower - Training

Fully Convolutional

$\mu(X,z)$

$\mu', \sigma'$

Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs
Decoder Tower - Training

P(Y|z, X)

Fully Convolutional

Output trajectories

Encoder Tower 8 Layers

Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs
Testing

Conditioned on Input Image

Sample from learnt distribution

Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs
Results

Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs
Results
Results
Video Demo

Cluster 4/5
15% of Samples

Video: http://www.cs.cmu.edu/~jcwalker/DTP/DTP.html
Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs
Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Negative Log Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regressor</td>
<td>11563</td>
</tr>
<tr>
<td>Optical Flow (Walker et al 2015)</td>
<td>11734</td>
</tr>
<tr>
<td>Proposed</td>
<td>11082</td>
</tr>
</tbody>
</table>

- Significantly outperforms all existing methods
Applications: Facial Expression Editing

- Instead of encoding pixels to a lower dimensional space, encode the flow.
- Uses bilinear sampling layer introduced in Spatial transformer networks (Covered in one of the previous lecture).

Disclaimer: I am one of the authors of this paper.
Single Image Expression Magnification and Suppression

Latent Space (z)

Image Credit: Semantic Facial Expression Editing Using Autoencoded Flow
Results: Expression Editing

Image Credit: Semantic Facial Expression Editing Using Autoencoded Flow
Results: Expression Interpolation

Latent Space (z)

These images in between are generated!

Image Credit: Semantic Facial Expression Editing Using Autoencoded Flow
GAN and VAEs are both popular
- Generative models use VAE for easy generation of $z$ given $X$.
- Generative models use GAN to generate sharp images given $z$.
- For images, model architecture follows DCGAN’s practices, using strided convolution, batch-normalization, and Relu.

Topics Not Covered:
Features learned from VAEs and GANs both can be used in the semi-supervised setting.
- “Semi-Supervised Learning with Deep Generative Models” [King ma et. al]
  (Follow up work by the original VAE author)
- “Auxiliary Deep Generative Models” [Maaløe, et. al]
Questions?
Reading List

- D. Kingma, M. Welling, Auto-Encoding Variational Bayes, ICLR, 2014
- Carl Doersch, Tutorial on Variational Autoencoders arXiv, 2016
- Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, Ole Winther, Autoencoding beyond pixels using a learned similarity metric, ICML, 2016
- Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, David Forsyth, Learning Diverse Image Colorization, arXiv, 2016

Not covered in this presentation:
- Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling, Semi-Supervised Learning with Deep Generative Models, NIPS, 2014