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

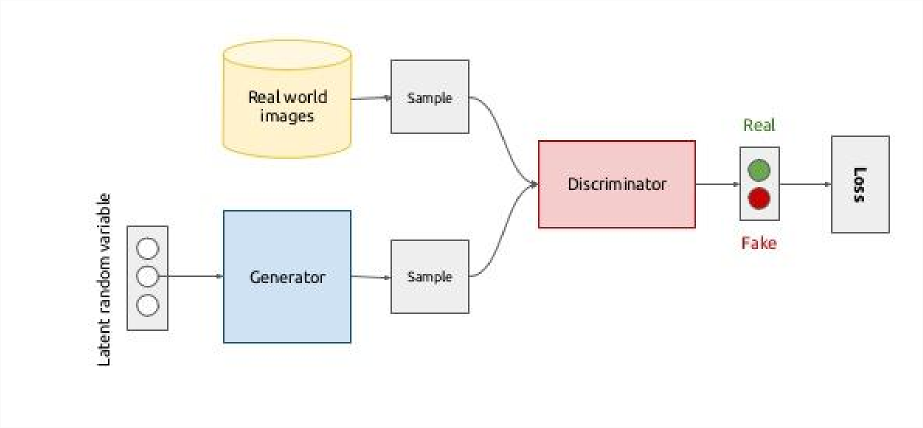
Recap: Generative Model + GAN

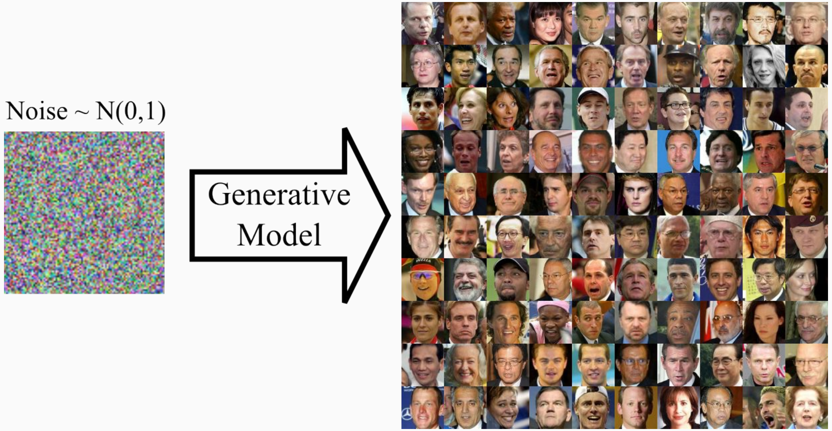
Last lecture we discussed **generative models**

- **Task:** Given a dataset of images {X1,X2...} 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.

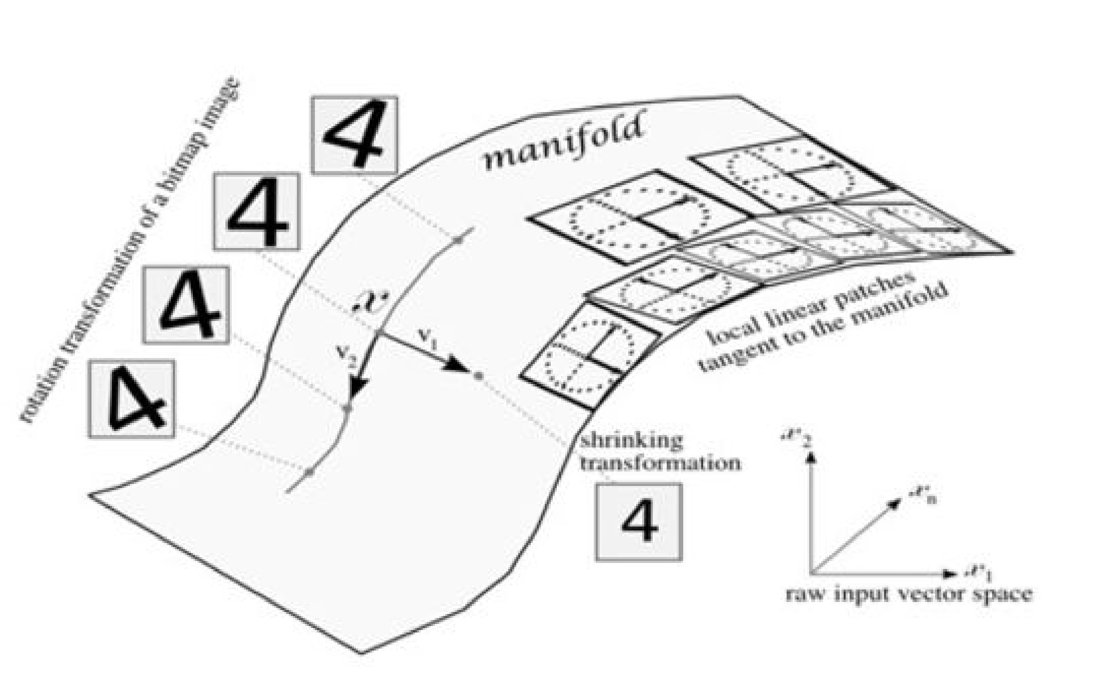


Image Credit: Deep learning book

Variational Autoencoder (VAE)

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

- Explicit Modelling of *P(X|z; θ)*, we will drop the θ in the notation.
- *z ~ 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 *θ* 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; θ)*, we will drop the θ in the notation.
- *z ~ 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 *θ* 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) ≈ 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 ~ 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) ≈ 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 ~ 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) ≈ 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 ~ Q(z) generates P(X|z) >> 0.

Variational Autoencoder (VAE)

Assume we can learn a distribution Q(z), where z ~ Q(z) generates P(X|z) >> 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 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

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

Variational Autoencoder (VAE)

Assume we can learn a distribution Q(z), where z ~ Q(z) generates P(X|z) >> 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 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

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

Variational Autoencoder (VAE)

Assume we can learn a distribution Q(z), where z ~ Q(z) generates P(X|z) >> 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 **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

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

Variational Autoencoder (VAE)

Assume we can learn a distribution Q(z), where z ~ Q(z) generates P(X|z) >> 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 **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

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

Variational Autoencoder (VAE)

Assume we can learn a distribution Q(z), where z ~ Q(z) generates P(X|z) >> 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 **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

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

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

- Definition of KL divergence:

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

- 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 $E_{z\sim Q}$

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

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

- Definition of KL divergence:

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

- 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 $E_{z\sim Q}$

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

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

- Definition of KL divergence:

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

- 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 $E_{z\sim Q}$

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

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

- 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:

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

- 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 $E_{z\sim Q}$

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

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

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

Rearrange the terms:

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

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

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

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

Rearrange the terms:

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

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

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

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

Rearrange the terms:

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

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

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

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

Rearrange the terms:

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

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

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

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

Rearrange the terms:

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

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

Intuition

Why is this important?

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

- Recall we want to maximize P(X) with respect to θ, 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?

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

- Recall we want to **maximize P(X)** with respect to θ, 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?

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

- Recall we want to **maximize P(X)** with respect to θ, 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?

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

- Recall we want to **maximize P(X)** with respect to θ, 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?

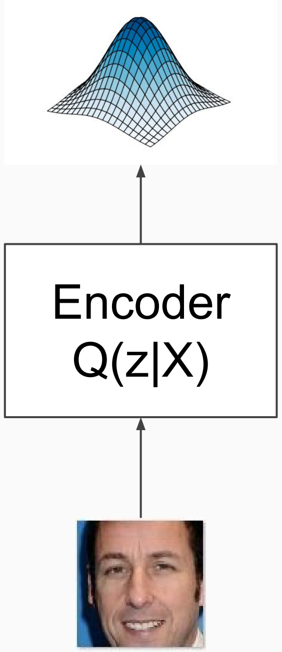
$$\boxed{\log P(X) - \boxed{\mathcal{D}\left[Q(z) \| P(z|X)\right]}} = E_{z\sim Q}\left[\log P(X|z)\right] - \mathcal{D}\left[\boxed{Q(z)} \| P(z)\right]$$

- Recall we want to **maximize P(X)** with respect to θ, 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) ?**

How to Get Q(z)?

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

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



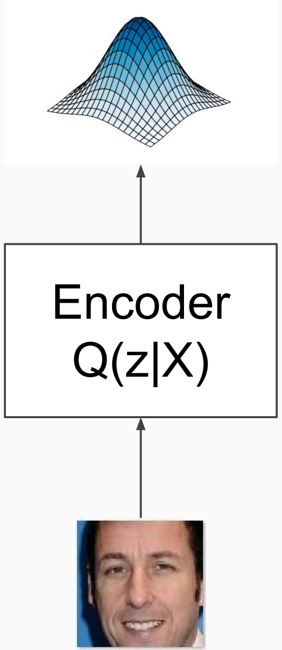
Let's call Q(z|X) the **Encoder.**

How to Get Q(z)?

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

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

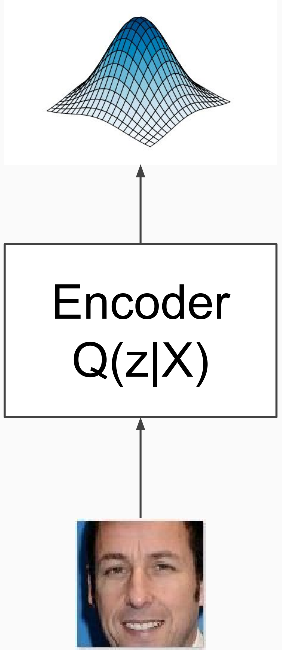
Let's call Q(z|X) the **Encoder.**



How to Get Q(z)?

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

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



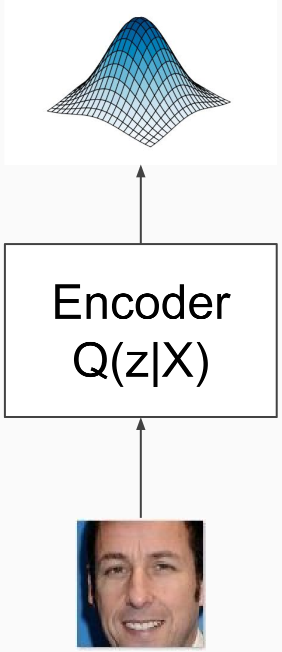
Let's call Q(z|X) the **Encoder.**

How to Get Q(z)?

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

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

Let's call Q(z|X) the **Encoder.**

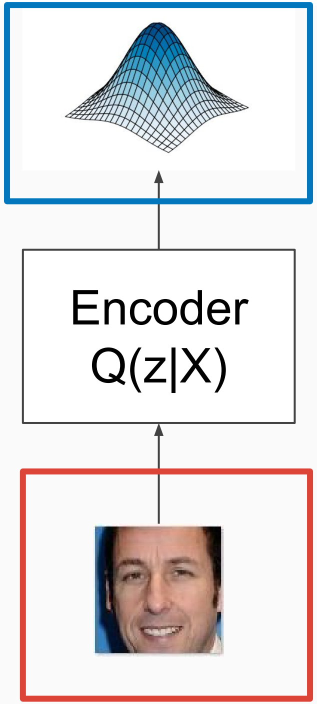


How to Get Q(z)?

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

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

Let’s call Q(z|X) the **Encoder.**

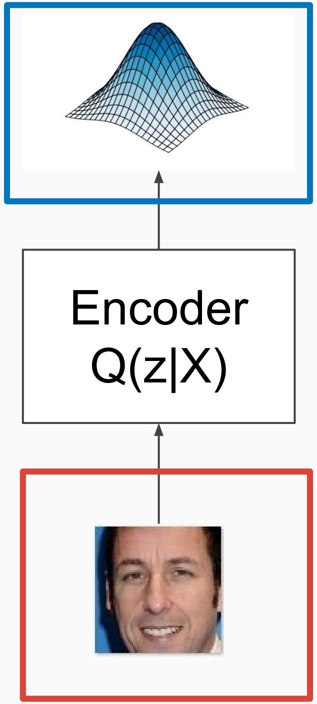


How to Get Q(z)?

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

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

Let’s call Q(z|X) the **Encoder.**



VAE's Loss function

Convert the lower bound to a loss function:

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

- 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:

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

- 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:

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

- 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:

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

- 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.

VAE's Loss function

Convert the lower bound to a loss function:

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

- 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) + η , where η ~ 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:

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

- 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) + η , where η ~ 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:

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

Assume P(z) ~ 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.

Pixel difference

Regularization

VAE's Loss function

Convert the lower bound to a loss function:

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

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

Putting it all together: $E_{z\sim Q(z|X)} \log P(X|z) \propto ||X-f(z)||^2$

$$L = \underbrace{||X - f(z)||^2}_{\textbf{Pixel difference}} - \underbrace{\lambda \cdot D[Q(z) \,||\, P(z)]}_{\textbf{Regularization}}$$

, given a (X, z) pair.

VAE's Loss function

Convert the lower bound to a loss function:

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

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.

Pixel difference

Regularization

VAE's Loss function

Convert the lower bound to a loss function:

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

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.

Pixel difference

Regularization

Variational Autoencoder

Training the Decoder is easy, just standard backpropagation.

How to train the Encoder?

- Not obvious how to apply gradient descent through samples.

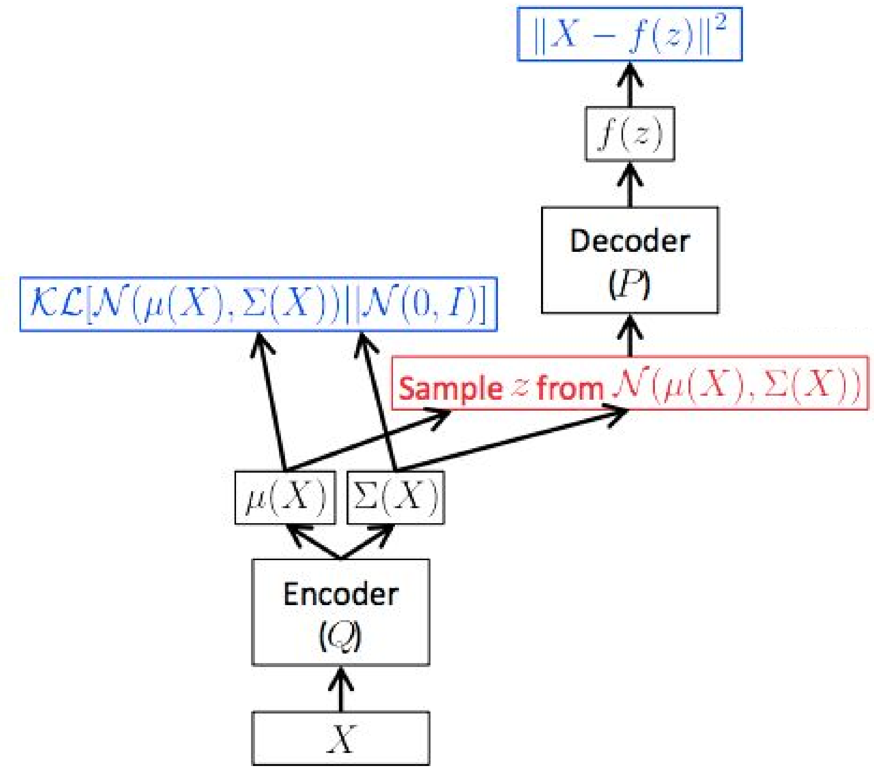


Image Credit: Tutorial on VAEs & unknown

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.

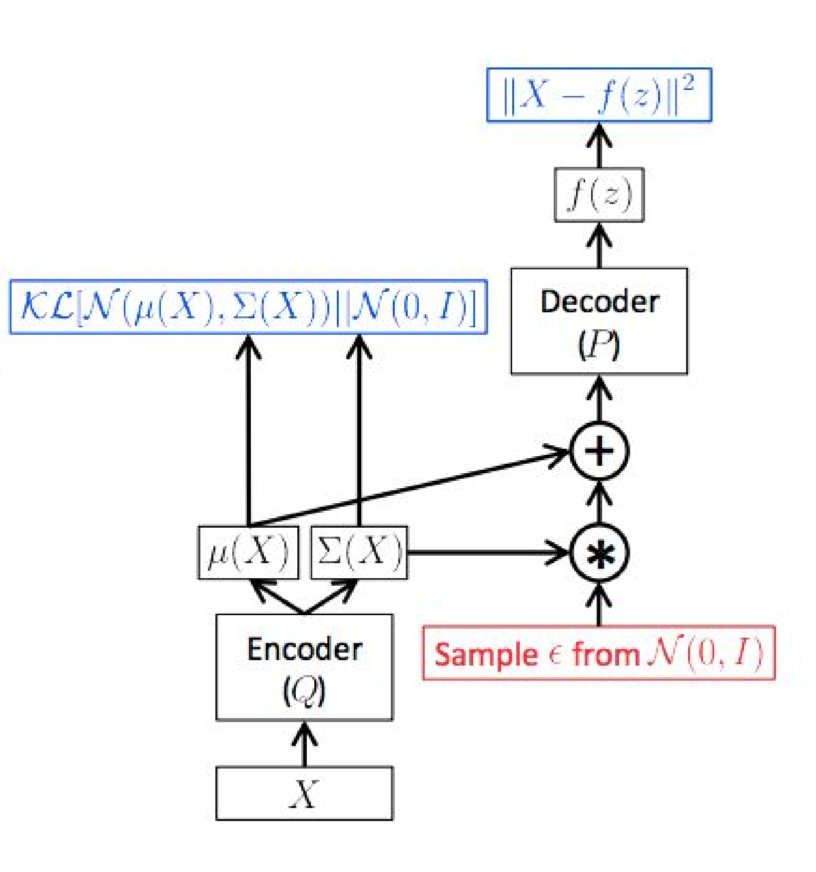


Image Credit: Tutorial on VAEs

VAE Training

Given a dataset of examples $\mathbf{X} = \{X1, X2...\}$

Initialize parameters for Encoder and Decoder

Repeat till convergence:

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

ε <-- Sample M noise vectors from $N(0, I)$

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

Gradient descent on L to updated Encoder and Decoder.

VAE Training

Given a dataset of examples **X** = {X1, X2...}

Initialize parameters for Encoder and Decoder

Repeat till convergence:

X^M <-- Random minibatch of M examples from X

ε <-- Sample M noise vectors from N(0, I)

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

Gradient descent on L to updated Encoder and Decoder.

VAE Training

Given a dataset of examples **X** = {X1, X2...}

Initialize parameters for Encoder and Decoder

Repeat till convergence:

$\mathbf{X}^M$ <-- Random minibatch of M examples from **X**

ε <-- Sample M noise vectors from N(0, I)

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

Gradient descent on L to updated Encoder and Decoder.

VAE Training

Given a dataset of examples **X** = {X1, X2...}

Initialize parameters for Encoder and Decoder

Repeat till convergence:

> $\mathbf{X}^M$ <-- Random minibatch of M examples from **X**
>
> ε <-- Sample M noise vectors from N(0, I)
>
> **Compute *L*(X^M, ε, θ) (i.e. run a forward pass in the neural network)**
>
> Gradient descent on *L* to updated Encoder and Decoder.

VAE Training

Given a dataset of examples **X** = {X1, X2...}

Initialize parameters for Encoder and Decoder

Repeat till convergence:

$\mathbf{X}^M$ <-- Random minibatch of M examples from **X**

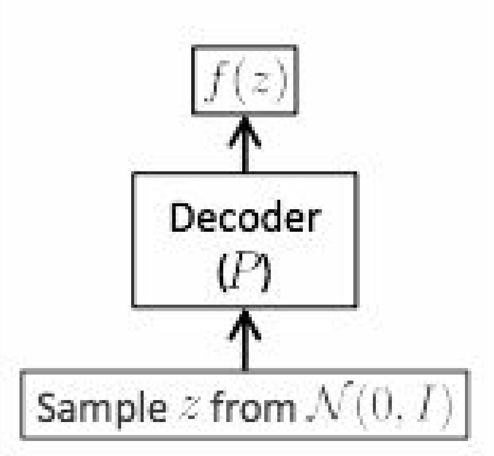
ε <-- Sample M noise vectors from N(0, I)

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

Gradient descent on *L* to updated Encoder and Decoder.

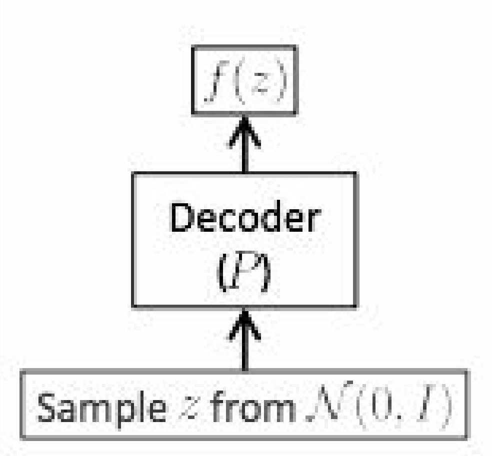
VAE Testing

- 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 ~ *N(0,I)* and pass it through the Decoder.
- No good quantitative metric, relies on visual inspection.



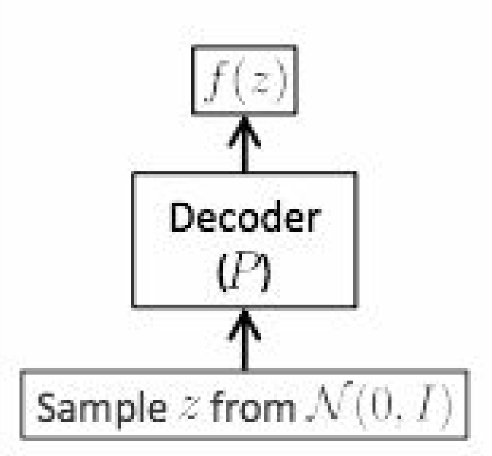
VAE Testing

- 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 ~ *N(0,I)* and pass it through the Decoder.
- No good quantitative metric, relies on visual inspection.



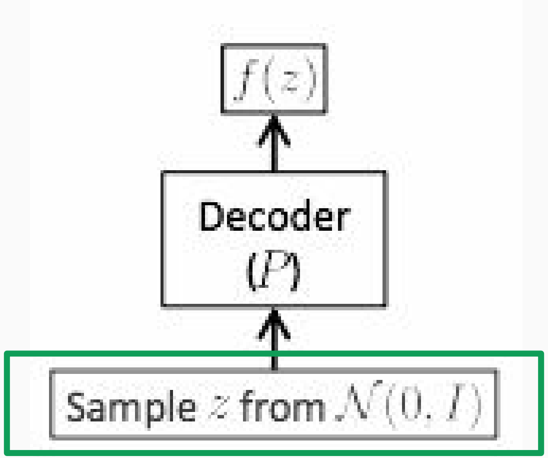
VAE Testing

- 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 ~ *N(0,I)* and pass it through the Decoder.
- No good quantitative metric, relies on visual inspection.



VAE Testing

- 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.



VAE Testing

- 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.**

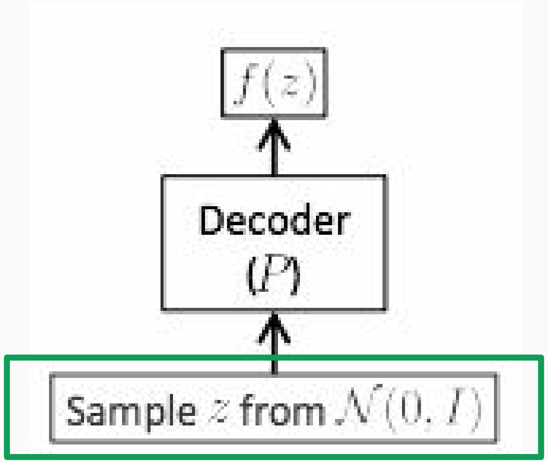
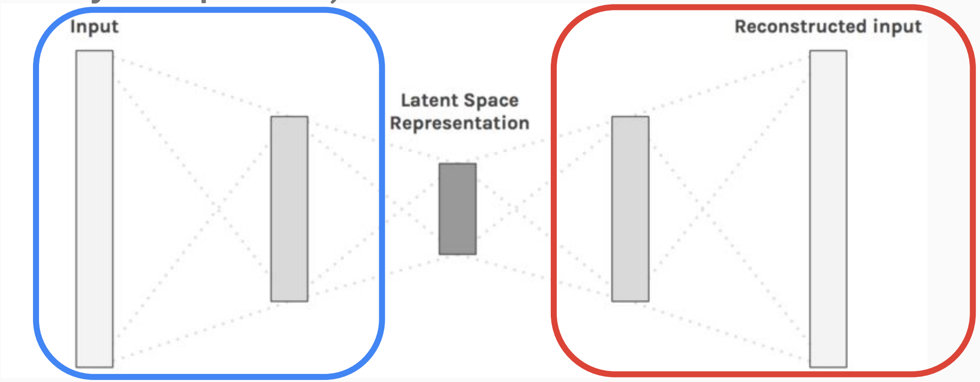


Image Credit: Tutorial on VAE

Common VAE architecture

Fully Connected (Initially Proposed)

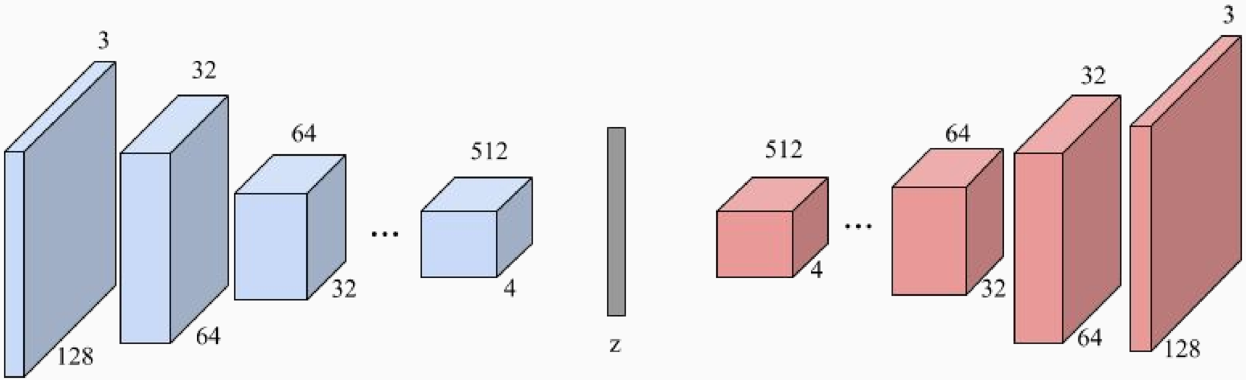
Encoder



Decoder

Common Architecture (convolutional) similar to DCGAN.

Encoder



Decoder

Disentangle latent factor

Autoencoder can disentangle latent factors [MNIST DEMO]:

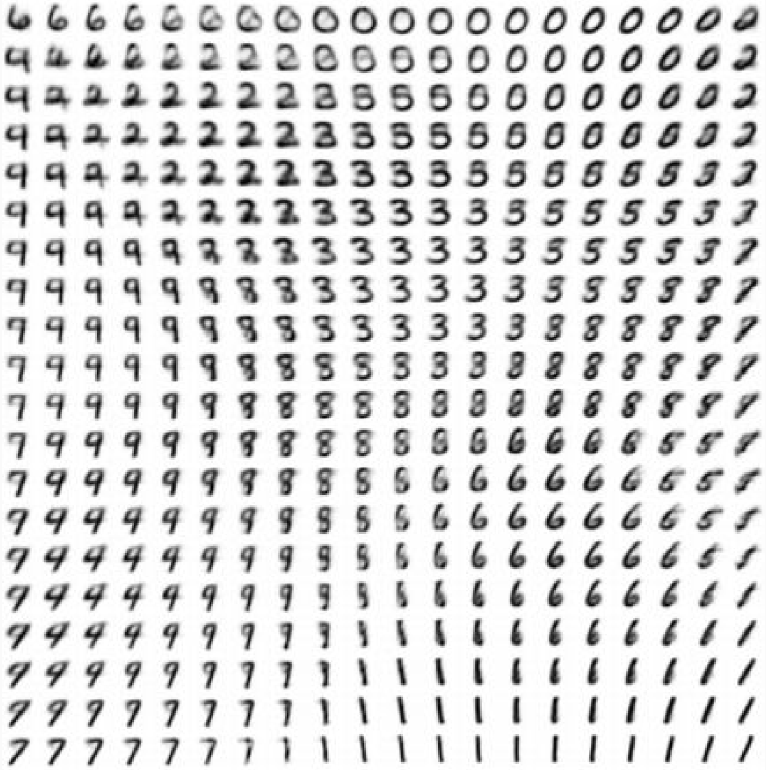
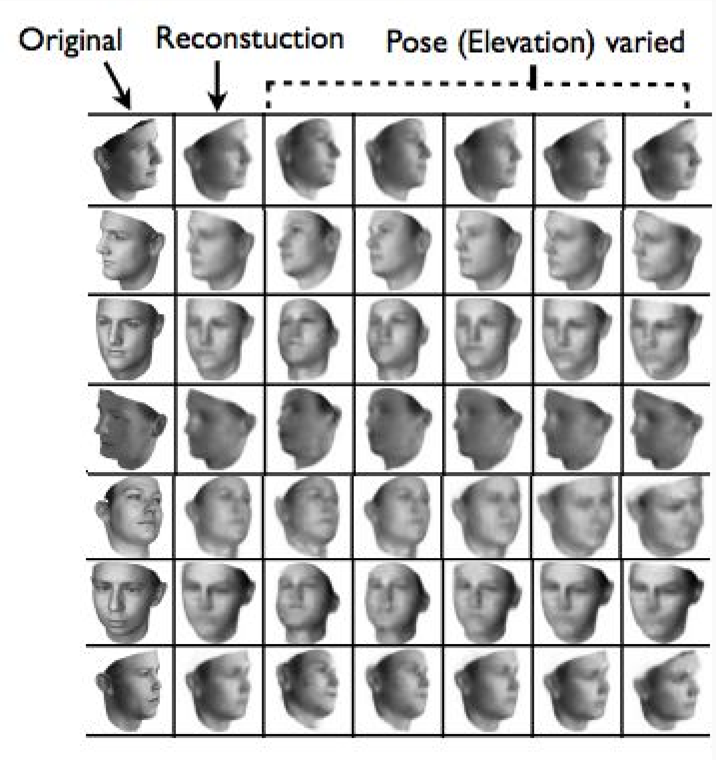


Image Credit: Auto-encoding Variational Bayes

Disentangle latent factor



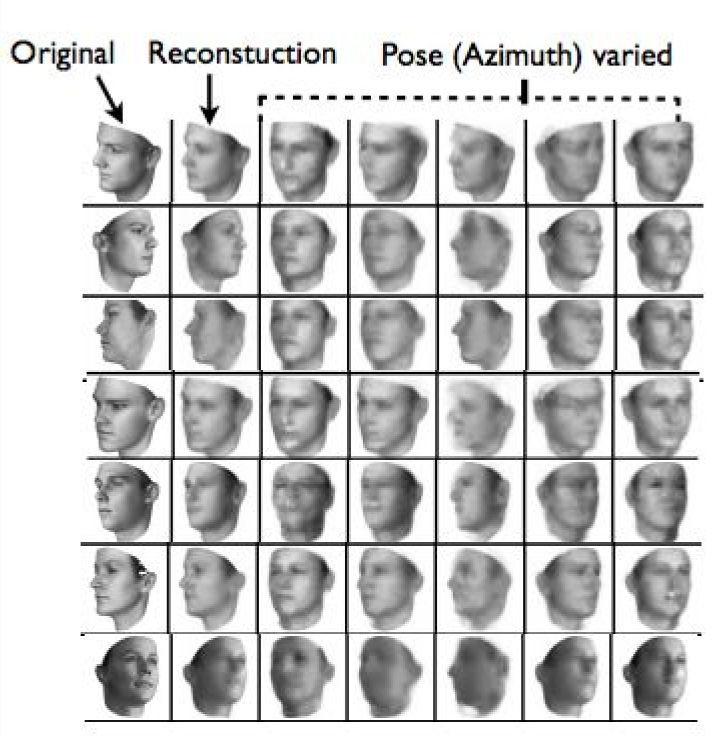
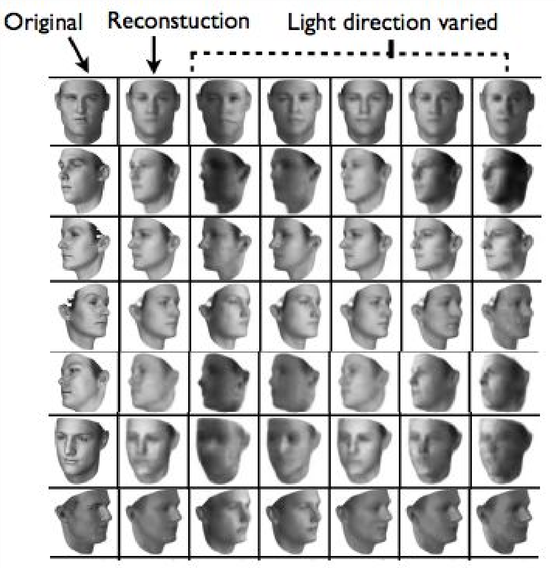


Image Credit: Deep Convolutional Inverse Graphics Network

Disentangle latent factor

We have seen **very similar results** during last lecture: InfoGan.

InfoGan



VAE

Image Credit: Deep Convolutional Inverse Graphics Network & InfoGan

VAE vs. GAN

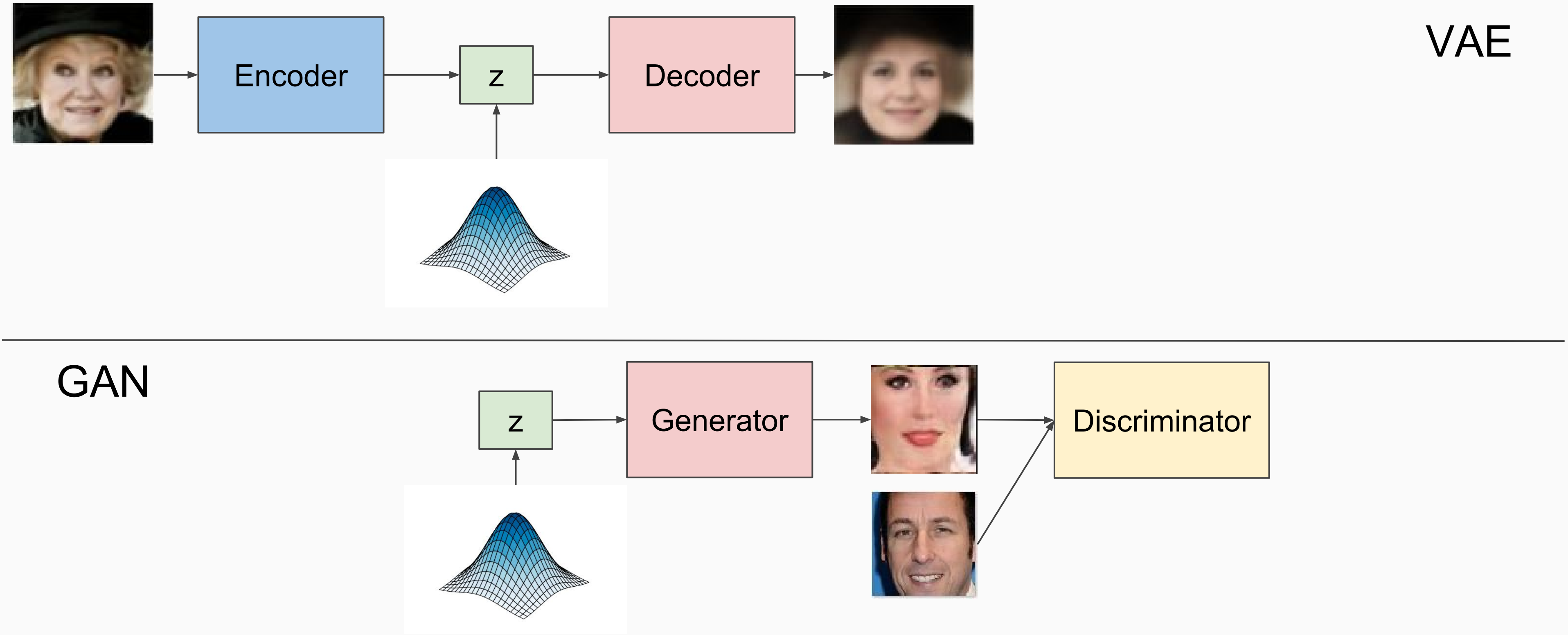
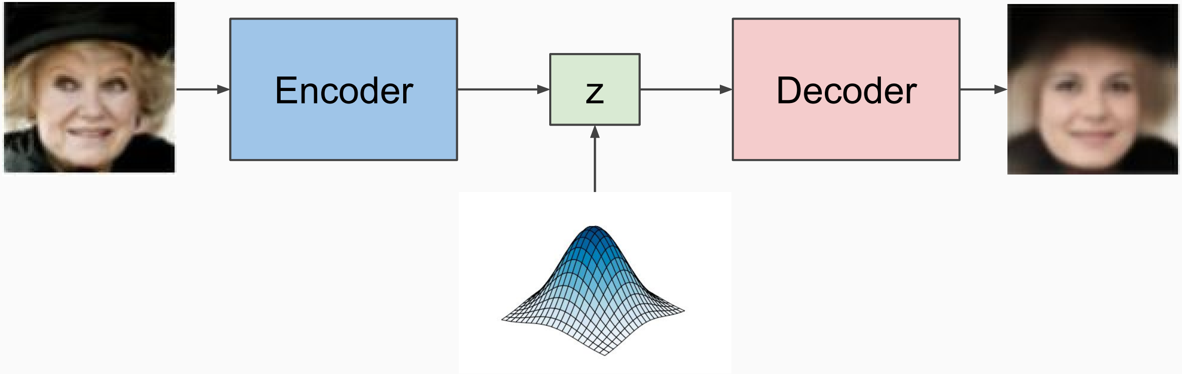


Image Credit: Autoencoding beyond pixels using a learned similarity metric

VAE vs. GAN

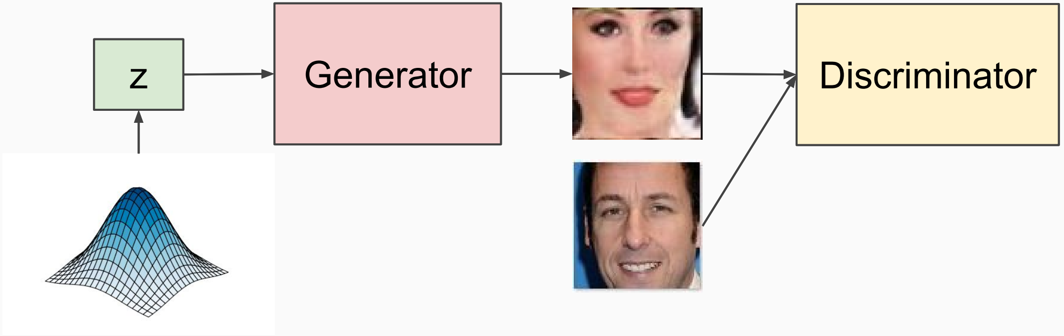


VAE

✓: Given an X **easy** to find z.
✓: Interpretable probability P(X)

X: Usually outputs blurry Images

GAN



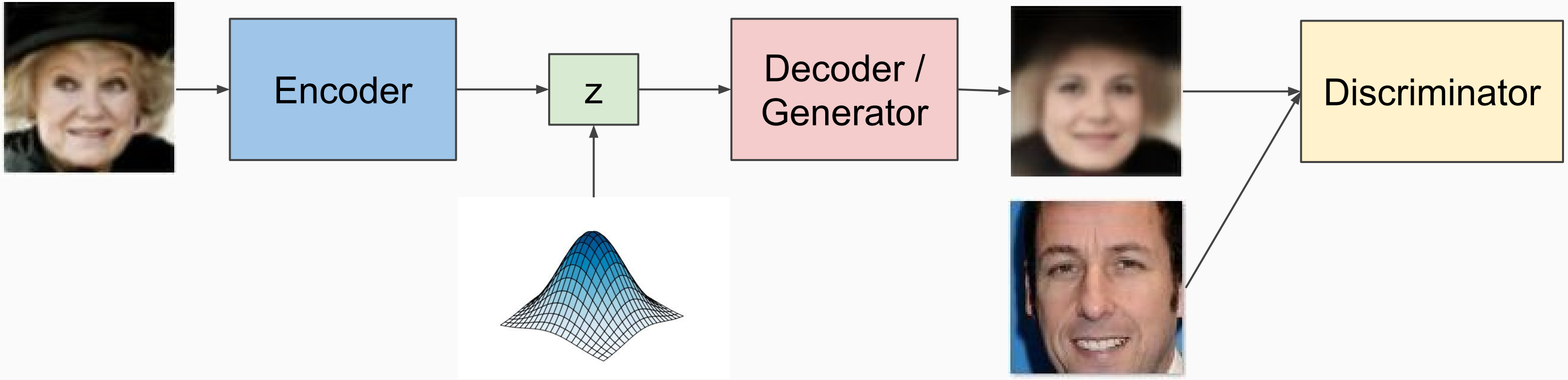
✓: Very sharp images

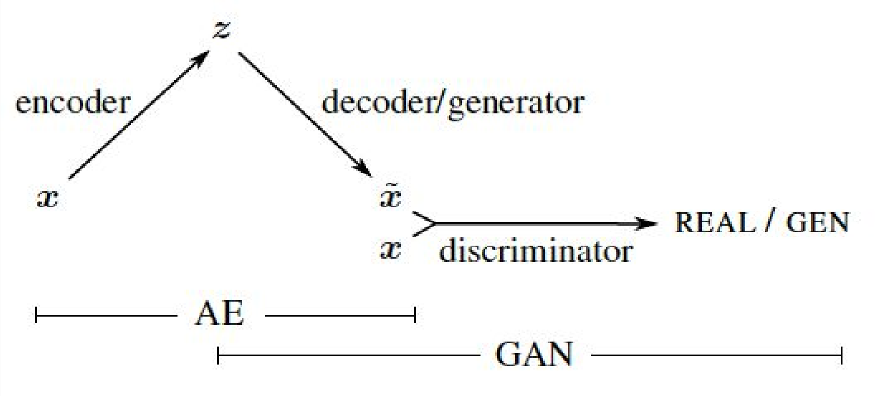
X: Given an X **difficult** to find z. (Need to backprop.)

✓/X: No explicit P(X).

Image Credit: Autoencoding beyond pixels using a learned similarity metric

GAN + VAE (Best of both models)





KL Divergence

L_2 Difference

$$\mathcal{L} = \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{llike}}^{\text{Dis}_l} + \mathcal{L}_{\text{GAN}}$$

Image Credit: Autoencoding beyond pixels using a learned similarity metric

Results

$VAE_{Dis/}$: Train a GAN first, then use the discriminator of GAN to train a VAE.

VAE/GAN: GAN and VAE trained together.

Image Credit: Autoencoding beyond pixels using a learned similarity metric

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.

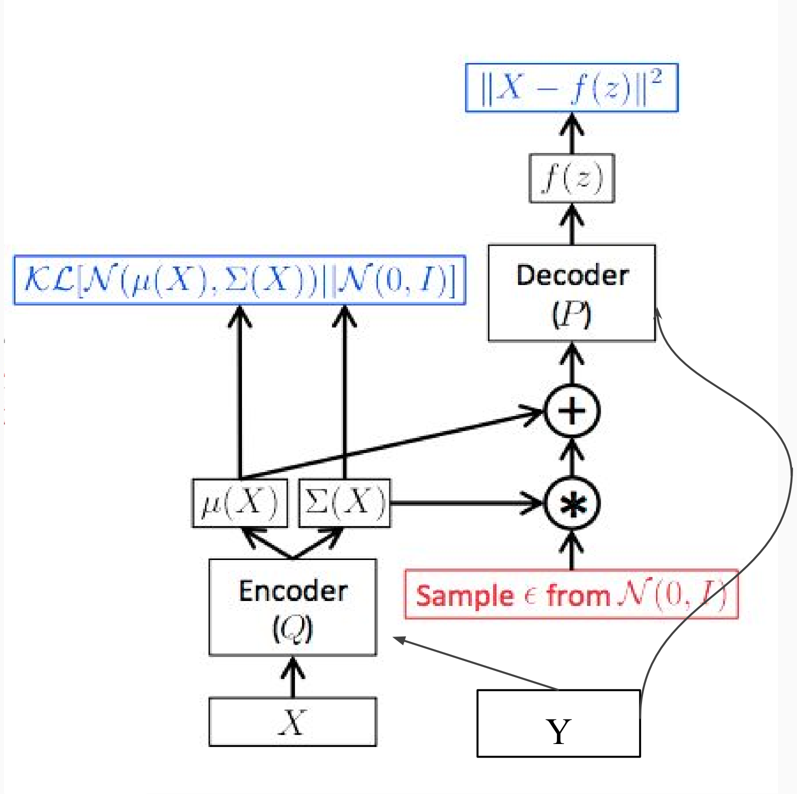


Image Credit: Tutorial on VAEs

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.

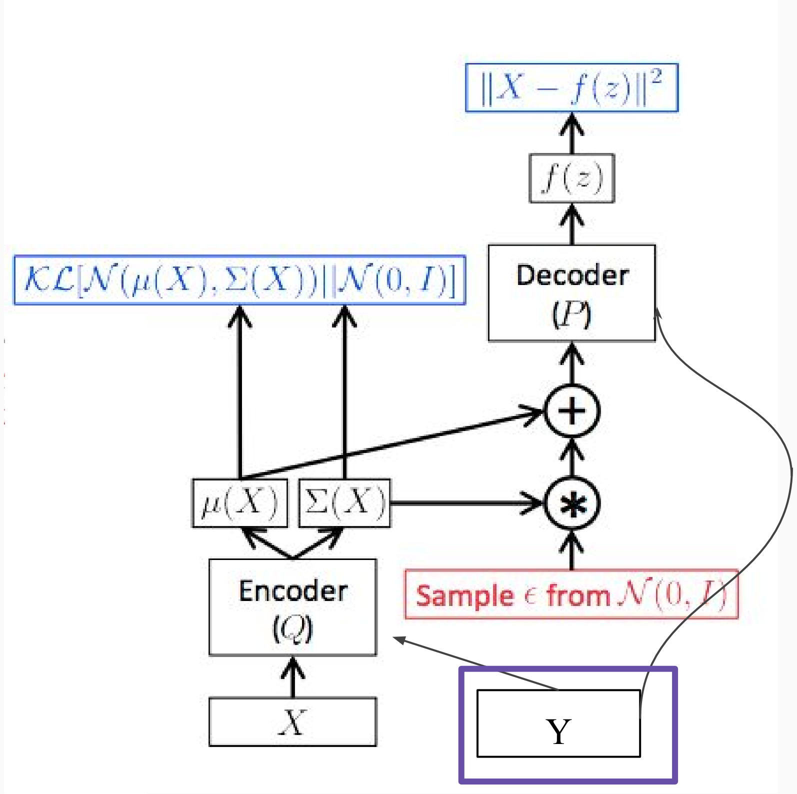


Image Credit: Tutorial on VAEs

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.

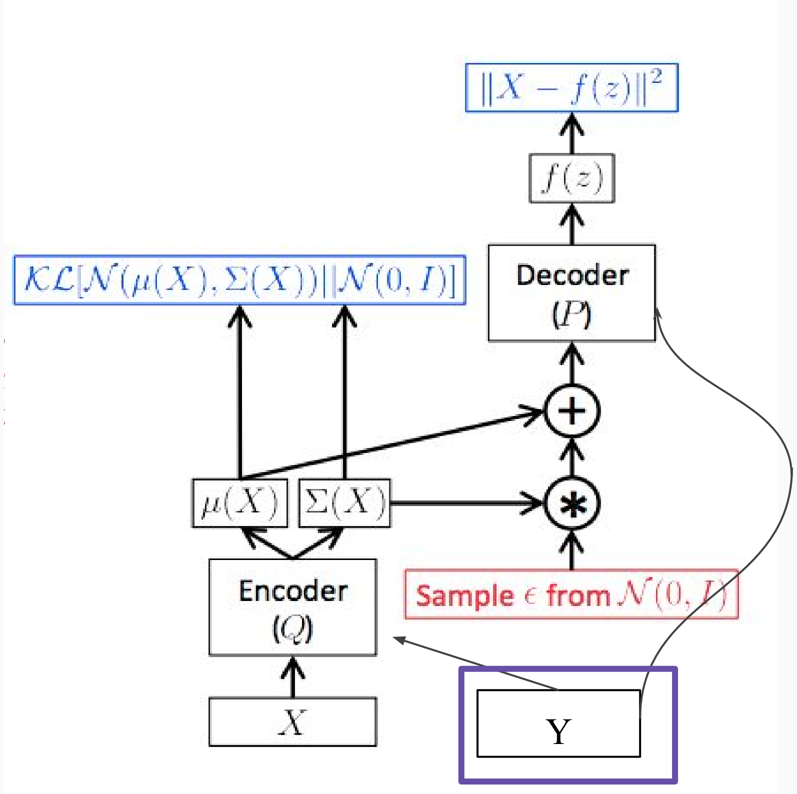
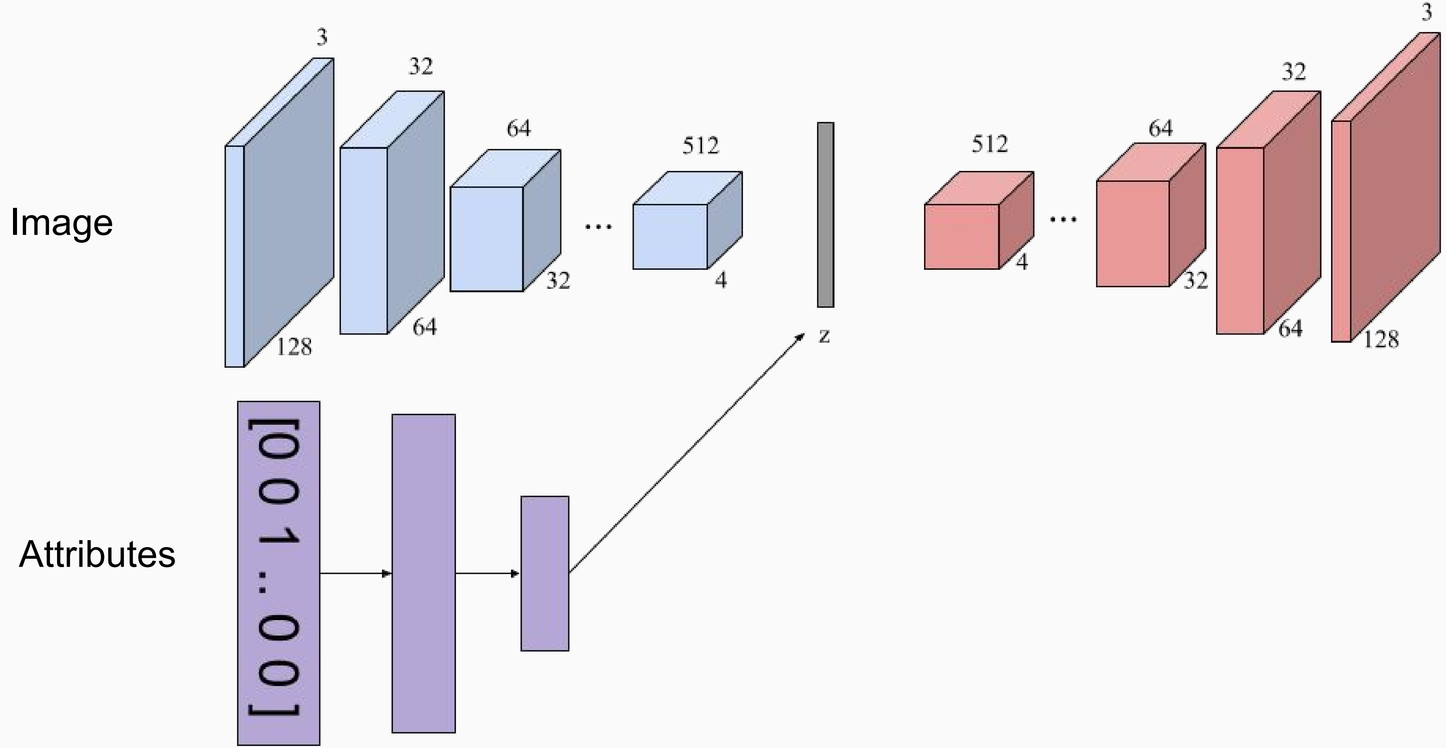


Image Credit: Tutorial on VAEs

Common CVAE architecture

Common Architecture (convolutional) for CVAE



CVAE Testing

- Again, remove the Encoder as test time
- Sample z ~ *N(0,I)* **and** input a **desired Y** to the Decoder.

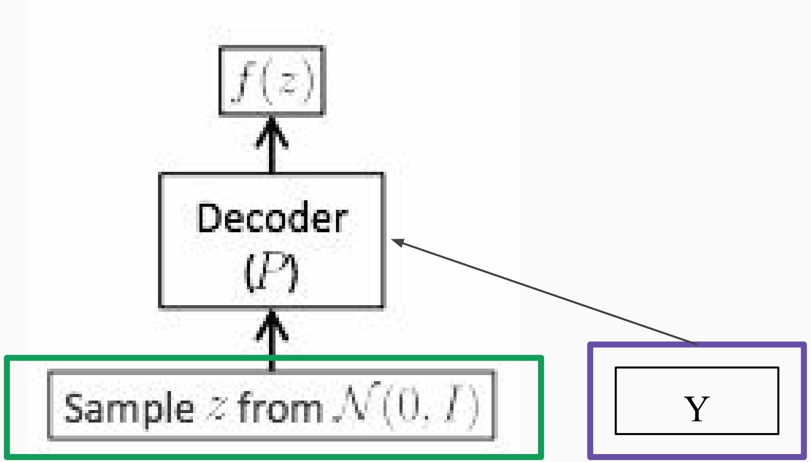


Image Credit: Tutorial on VAE

Example

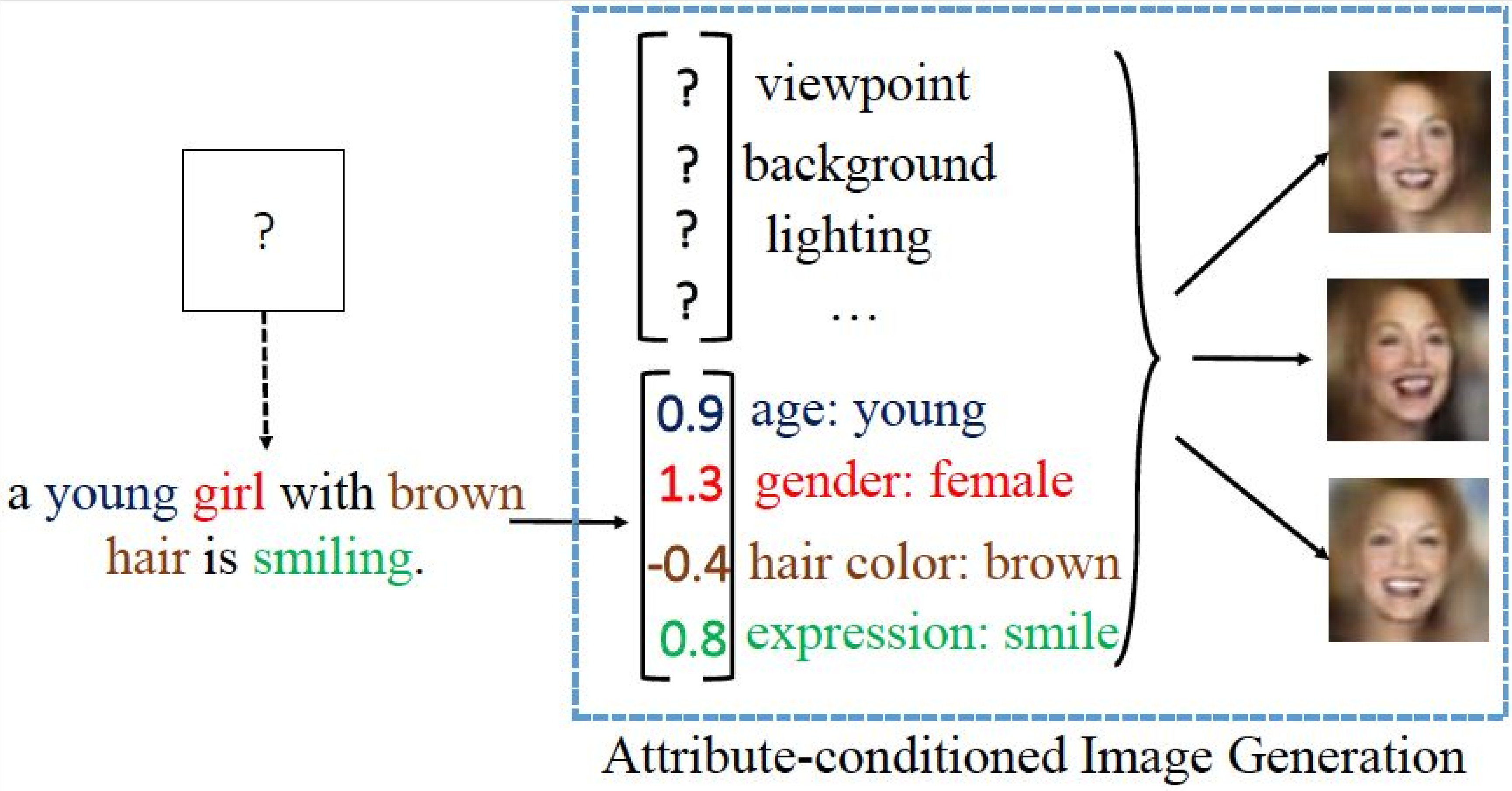
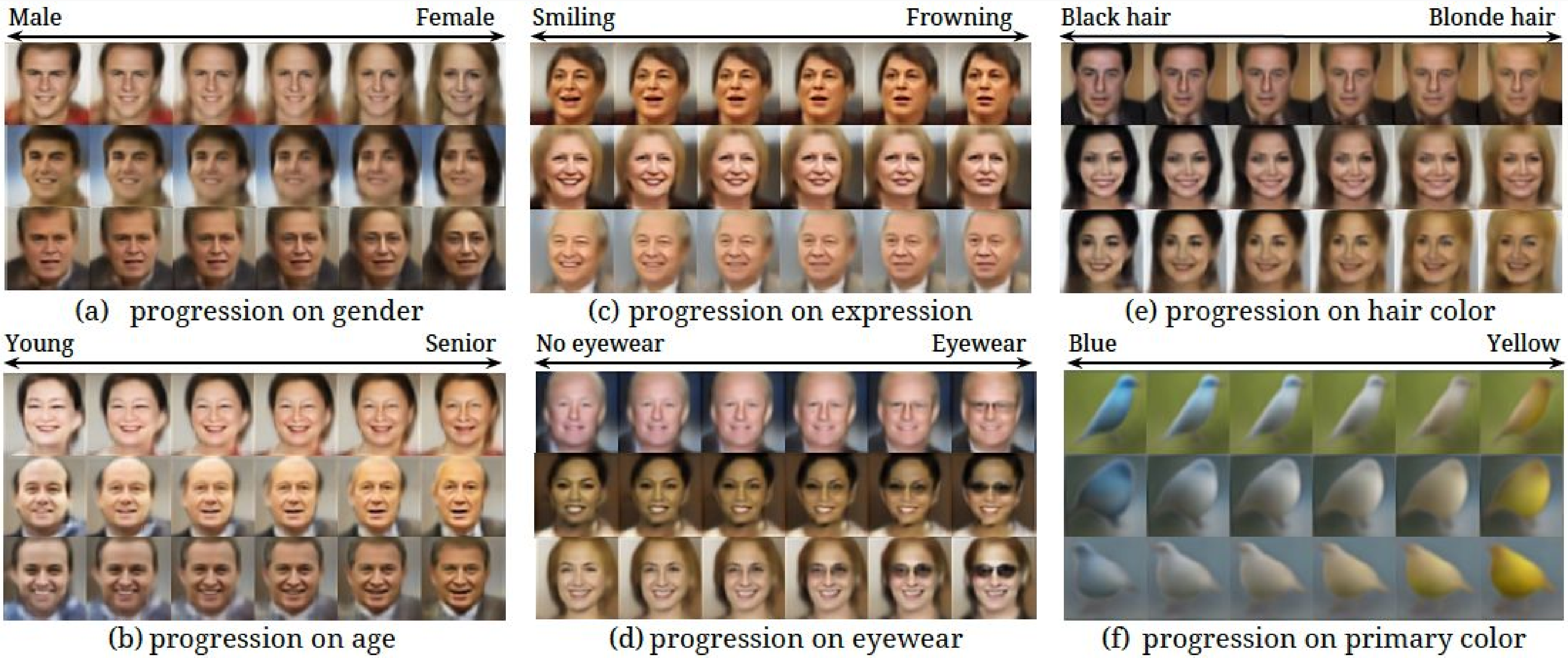


Image Credit: Attribute2Image

Attribute-conditioned image progression



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

Image Credit: Attribute2Image

Learning Diverse Image Colorization

Image Colorization

- An ambiguous problem

Picture Credit: https://pixabay.com/en/vw-camper-vintage-car-vw-vehicle-1939343/

Learning Diverse Image Colorization

Image Colorization

- An ambiguous problem

Blue?
Red?
Yellow?

Picture Credit: https://pixabay.com/en/vw-camper-vintage-car-vw-vehicle-1939343/

Strategy

Goal:

Learn a conditional model P(C|G)

Color field C, given grey level image G

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

Strategy

Goal:

Learn a conditional model P(C|G)

Color field C, given grey level image G

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

Difficult to learn!

Exceedingly high dimensions!
(Curse of dimensionality)

Strategy

Goal:

Learn a conditional model P(C|G)

Color field C, given grey level image G.

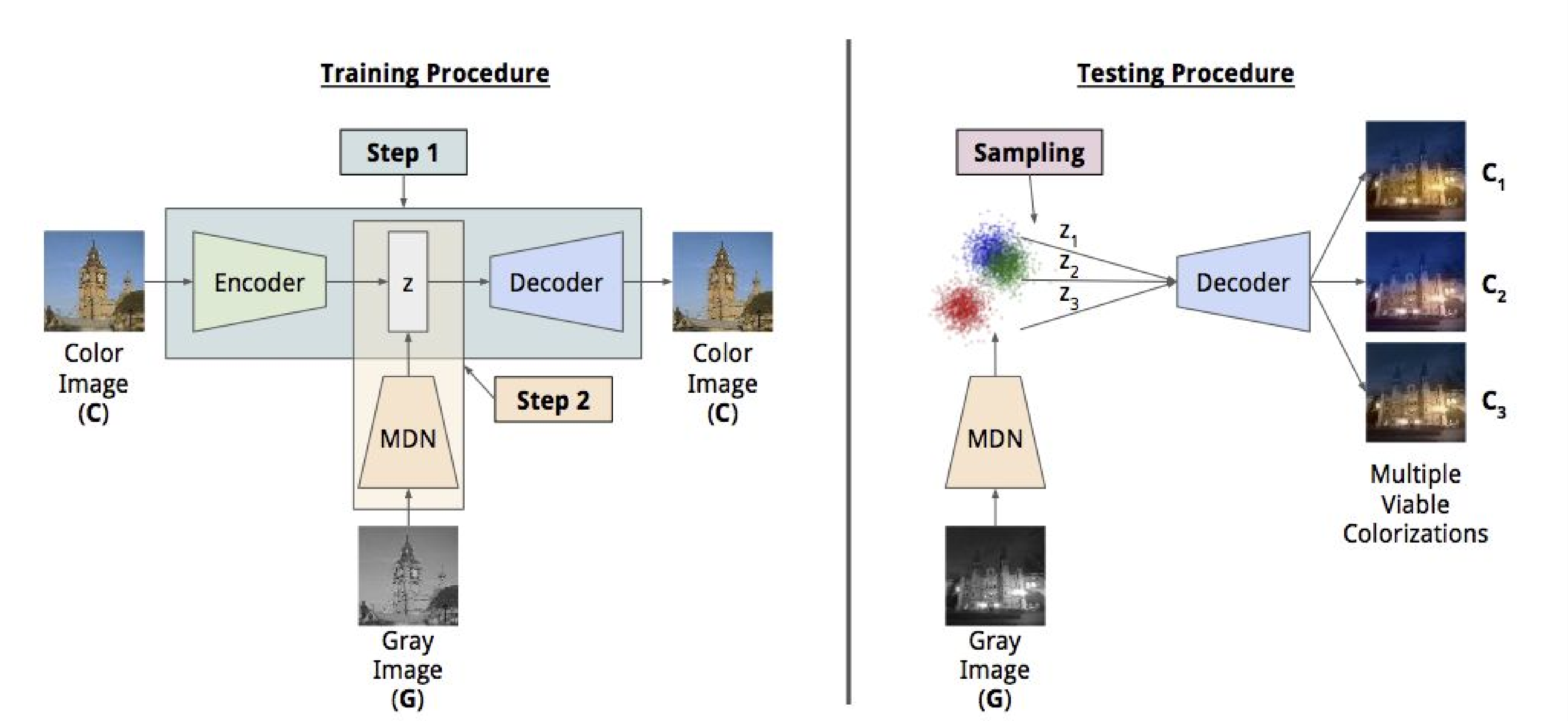
Instead of learning C directly, learn a low-dimensional embedding variable z (VAE).

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



[Image Credit: Learning Diverse Image Colorization]()

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.

$$\mathcal{L}_{dec} = \boxed{\mathcal{L}_{hist}} + \lambda_{mah}\mathcal{L}_{mah} + \lambda_{grad}\mathcal{L}_{grad}$$

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.

$$\mathcal{L}_{dec} = \boxed{\mathcal{L}_{hist}} + \lambda_{mah}\boxed{\mathcal{L}_{mah}} + \lambda_{grad}\mathcal{L}_{grad}$$

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.

$$\mathcal{L}_{dec} = \boxed{\mathcal{L}_{hist}} + \lambda_{mah}\boxed{\mathcal{L}_{mah}} + \lambda_{grad}\boxed{\mathcal{L}_{grad}}$$

Devil is in the details

Step 2: Conditional Model: Grey-level to Embedding

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

- 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.

Results

(a) Ours (b) CVAE (c) Zhang et al.[30] (d) GT

Image Credit: Learning Diverse Image Colorization

Effects of Loss Terms

Image Credit: Learning Diverse Image Colorization

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

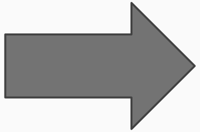


Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs

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

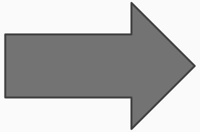


Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs

Applications: Forecasting from Static Images

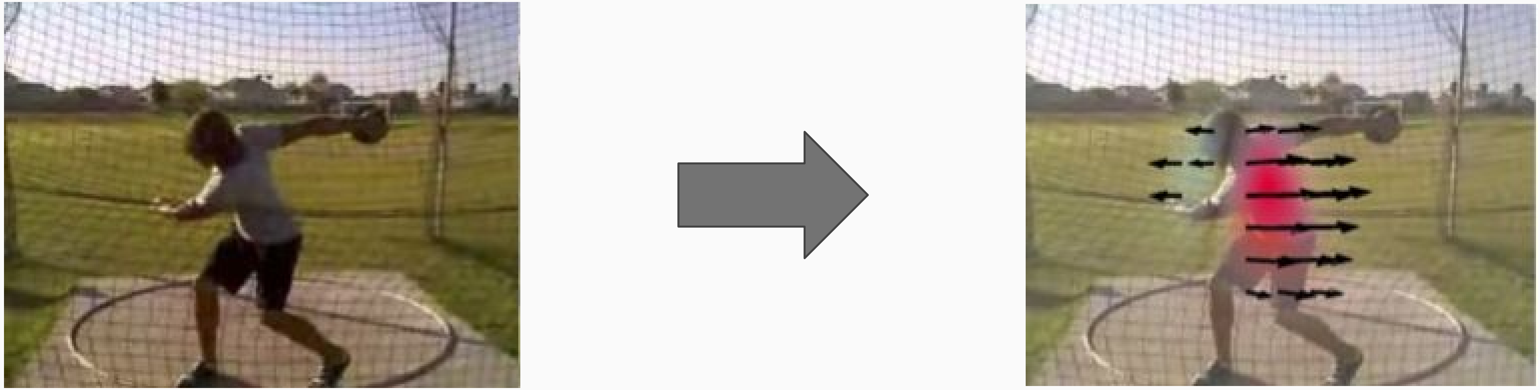


Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs

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

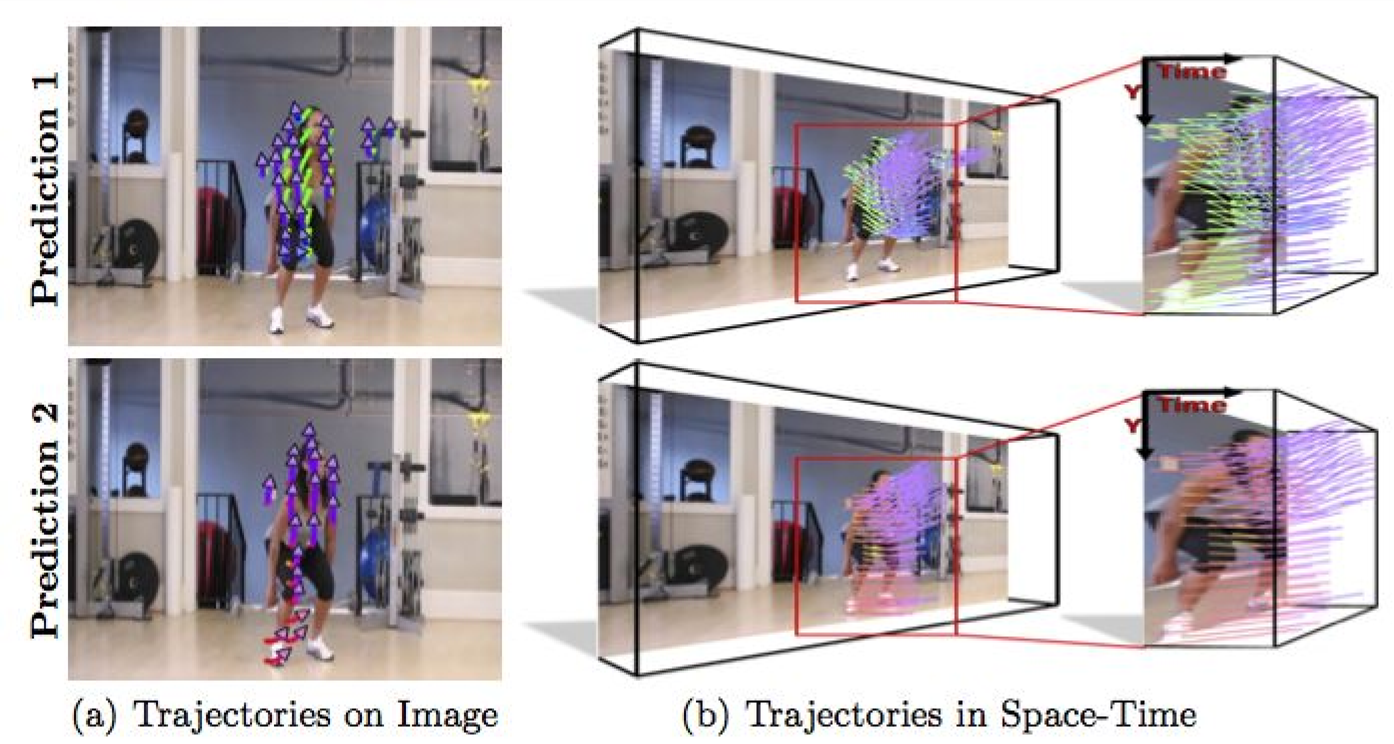
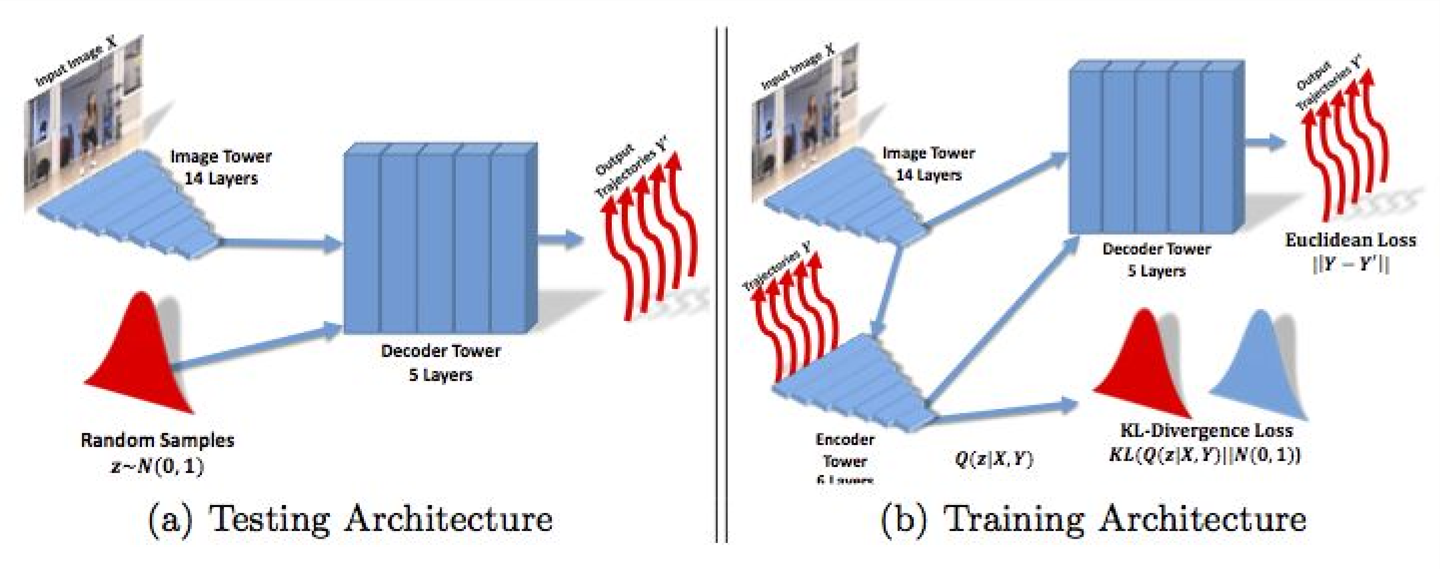


Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs

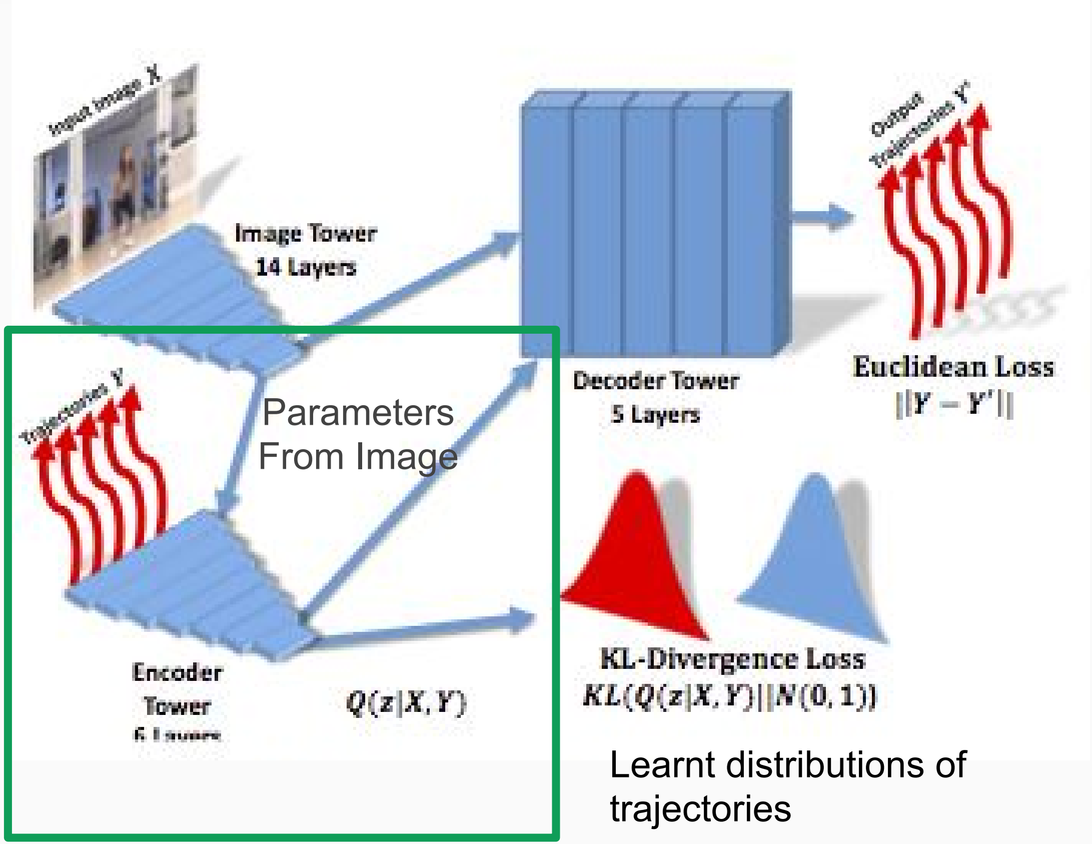
Architecture



[Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs]()

Encoder Tower - Training Only

Computed Optical Flow



Learnt distributions of trajectories

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

Image Tower - Training

Fully Convolutional

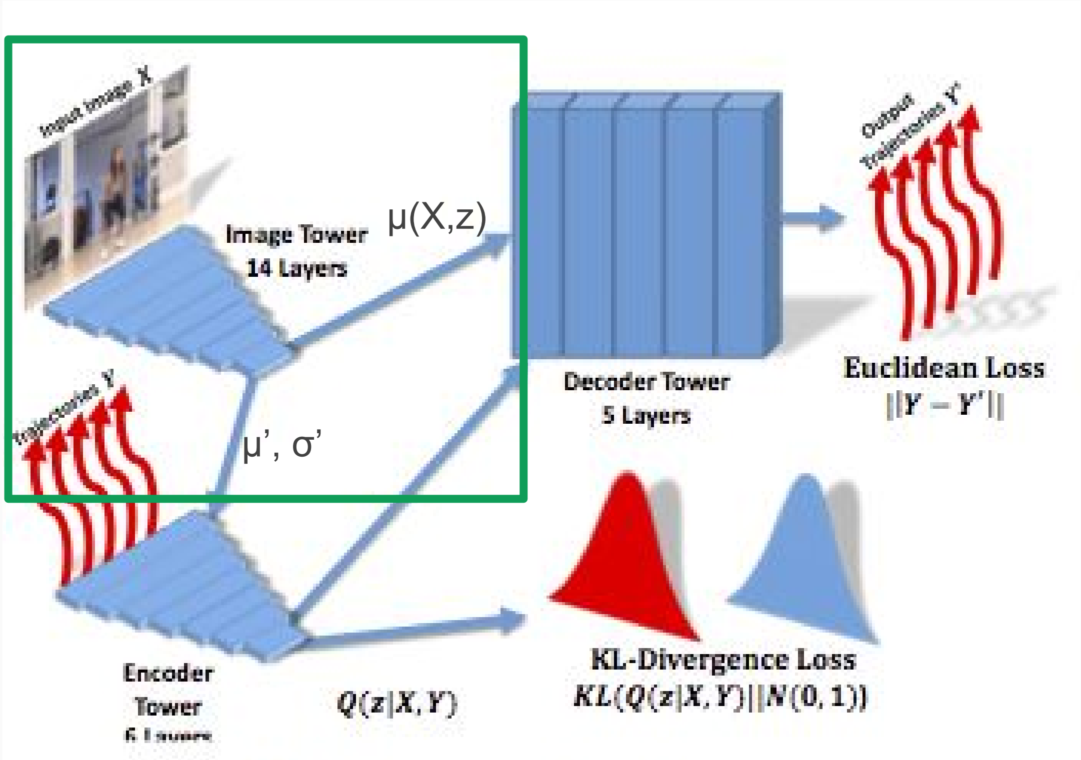
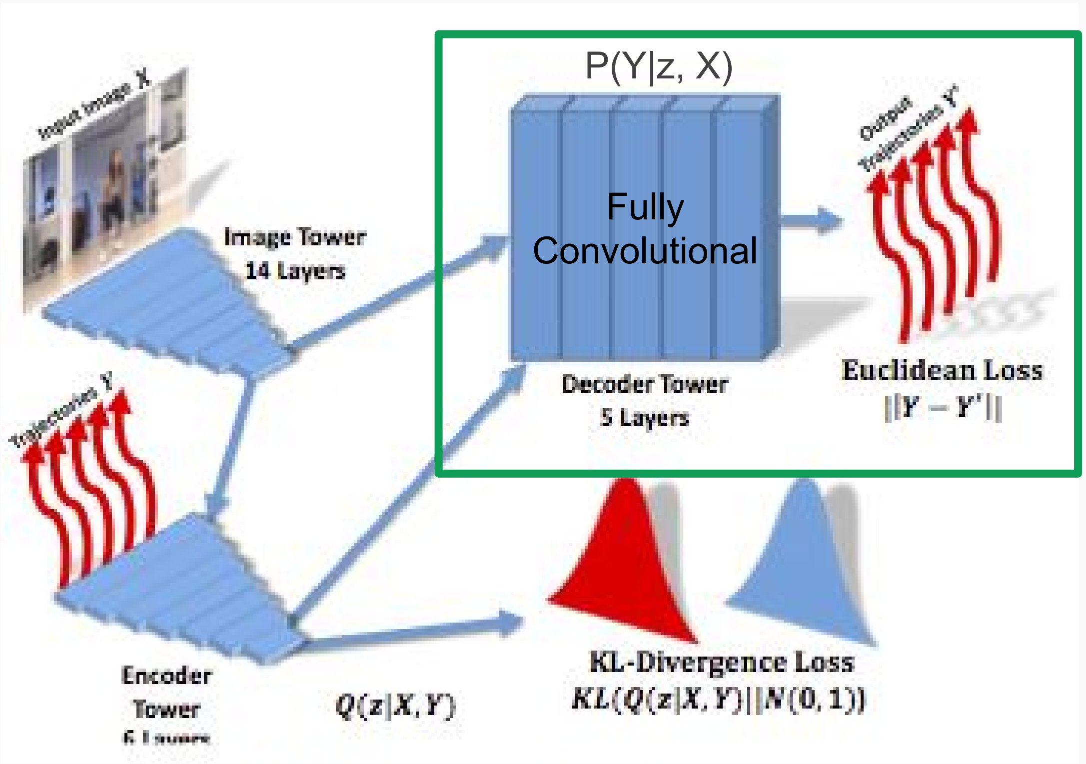


Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs

Decoder Tower - Training



Output trajectories

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

Testing

Conditioned on Input Image

Sample from learnt distribution

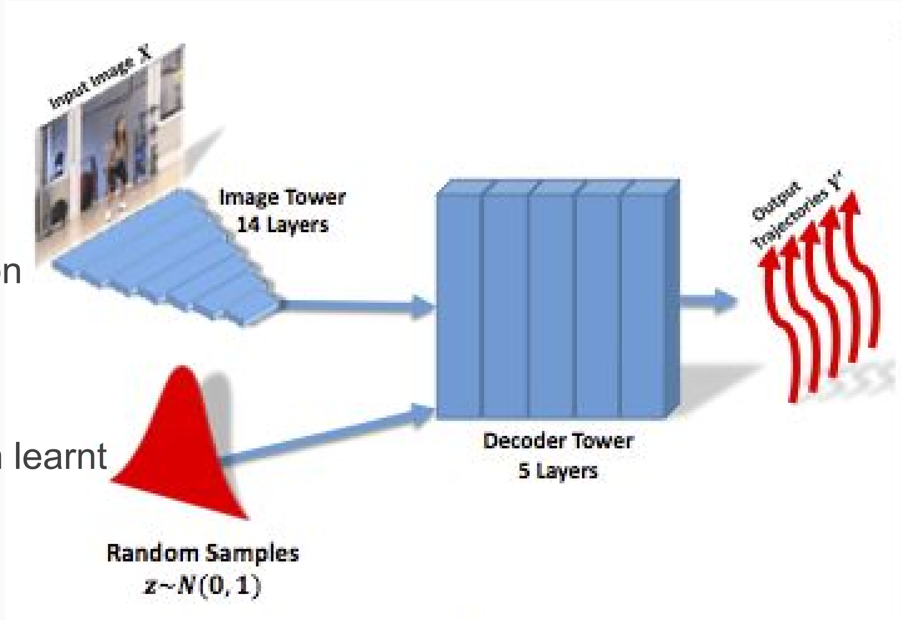


Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs

Results

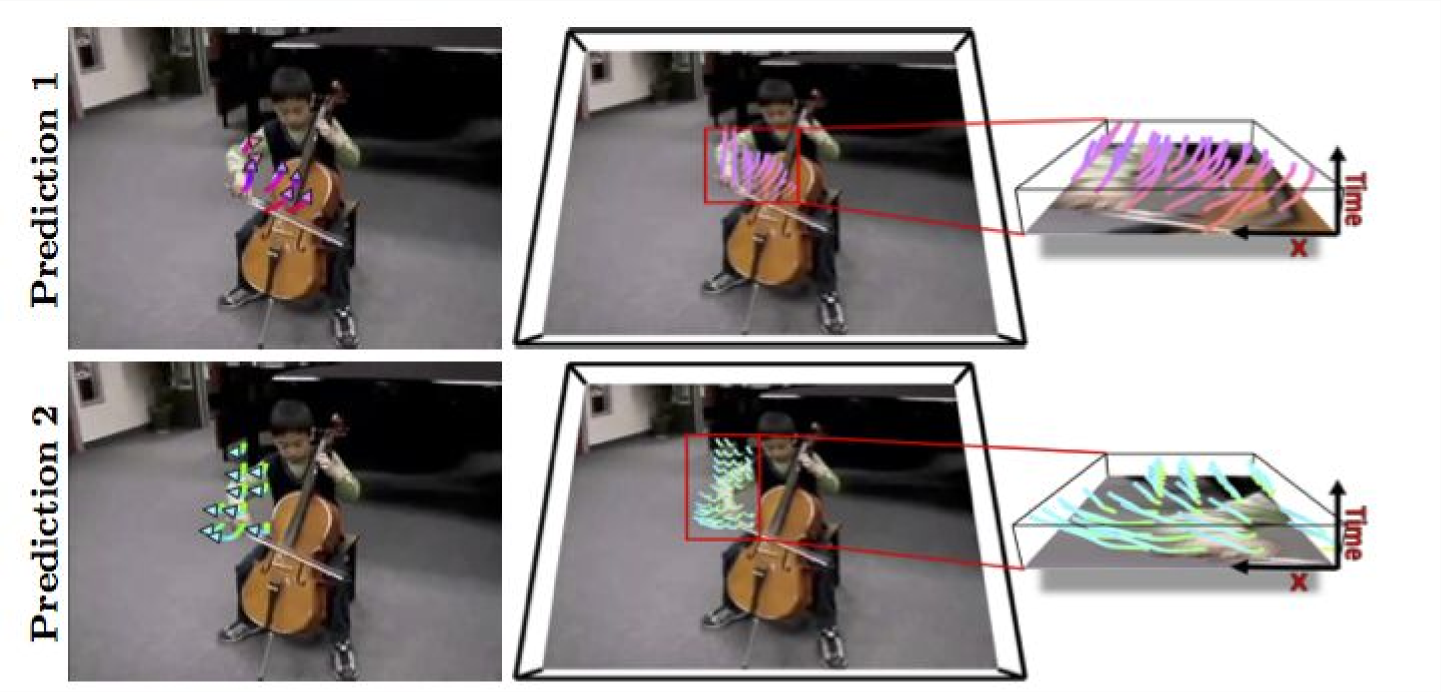


Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs

Results

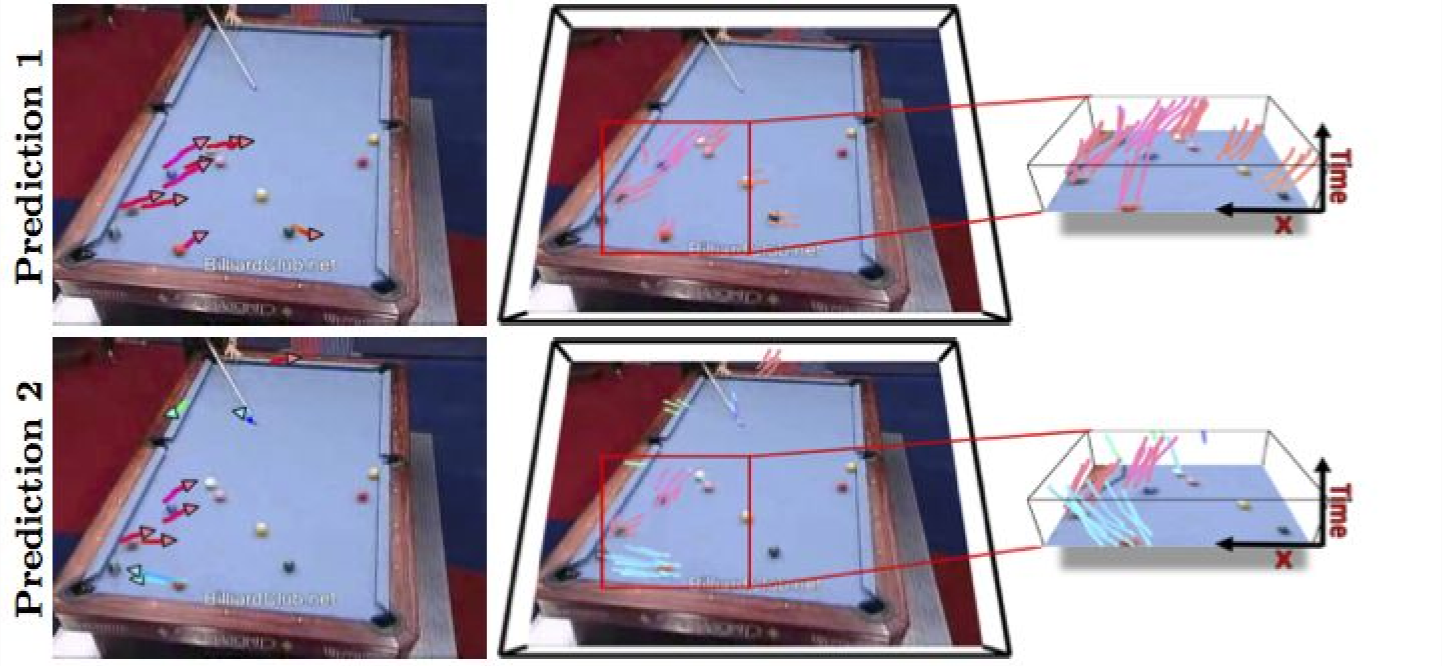


Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs

Results

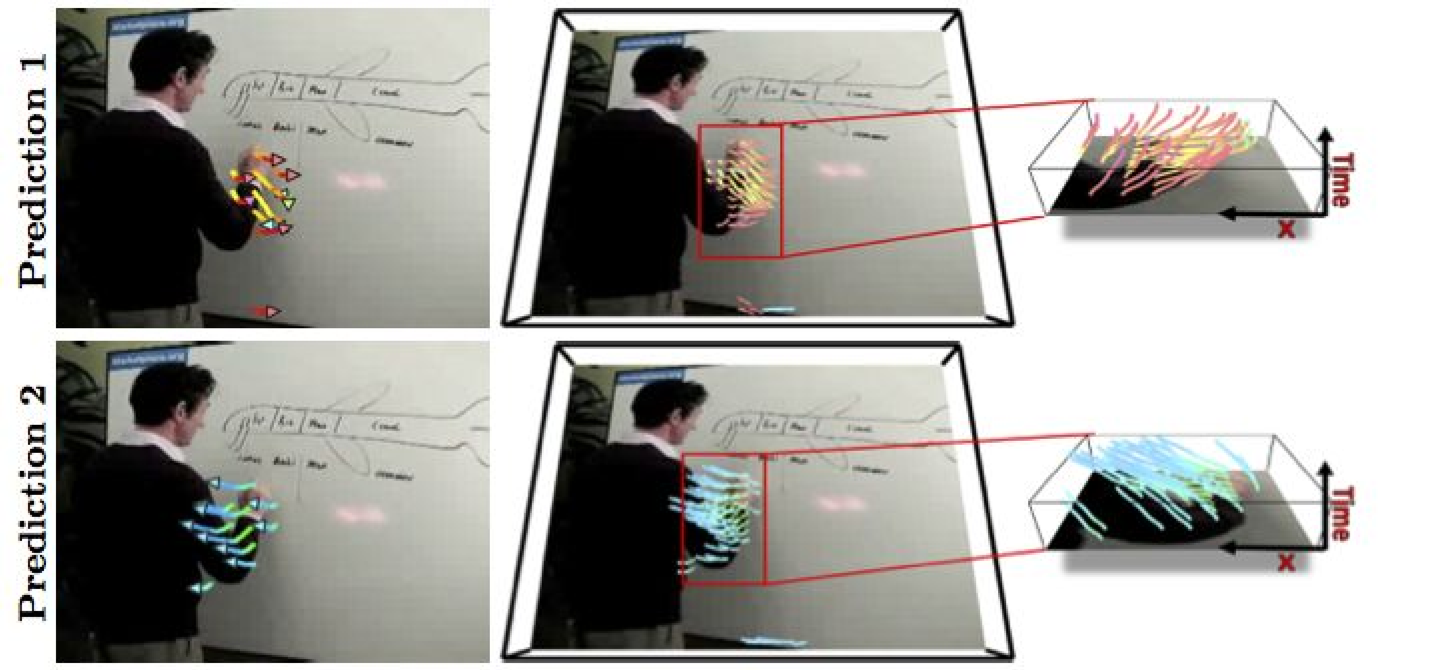


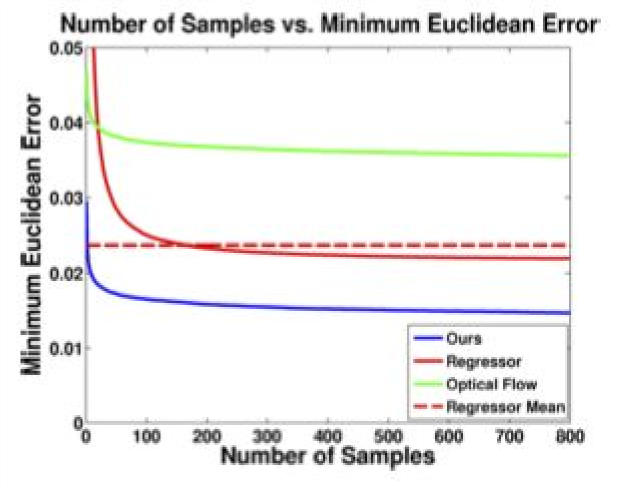
Image Credit: An Uncertain Future: Forecasting from static Images Using VAEs

Video Demo

Video: http://www.cs.cmu.edu/~jcwalker/DTP/DTP.html

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

Results

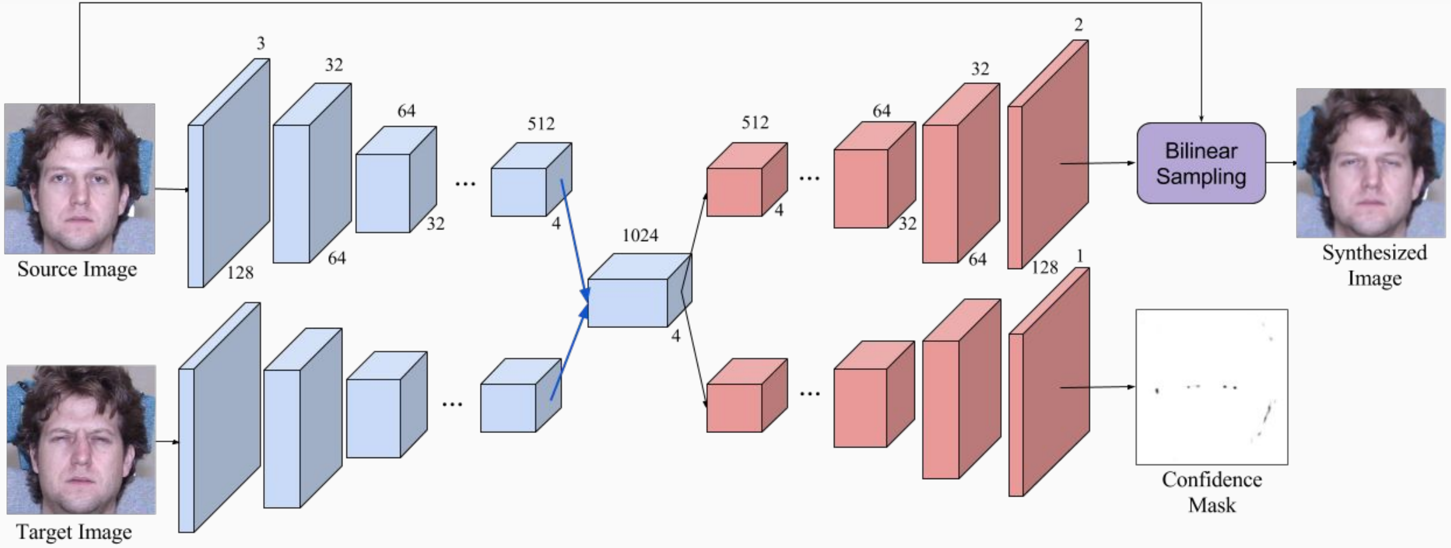


Method	Negative Log Likelihood
Regressor	11563
Optical Flow (Walker et al 2015)	11734
Proposed	**11082**

- Significantly outperforms all existing methods

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

Applications: Facial Expression Editing



Disclaimer: I am one of the authors of this paper.

- 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).

Image Credit: Semantic Facial Expression Editing Using Autoencoded Flow

Single Image Expression Magnification and Suppression

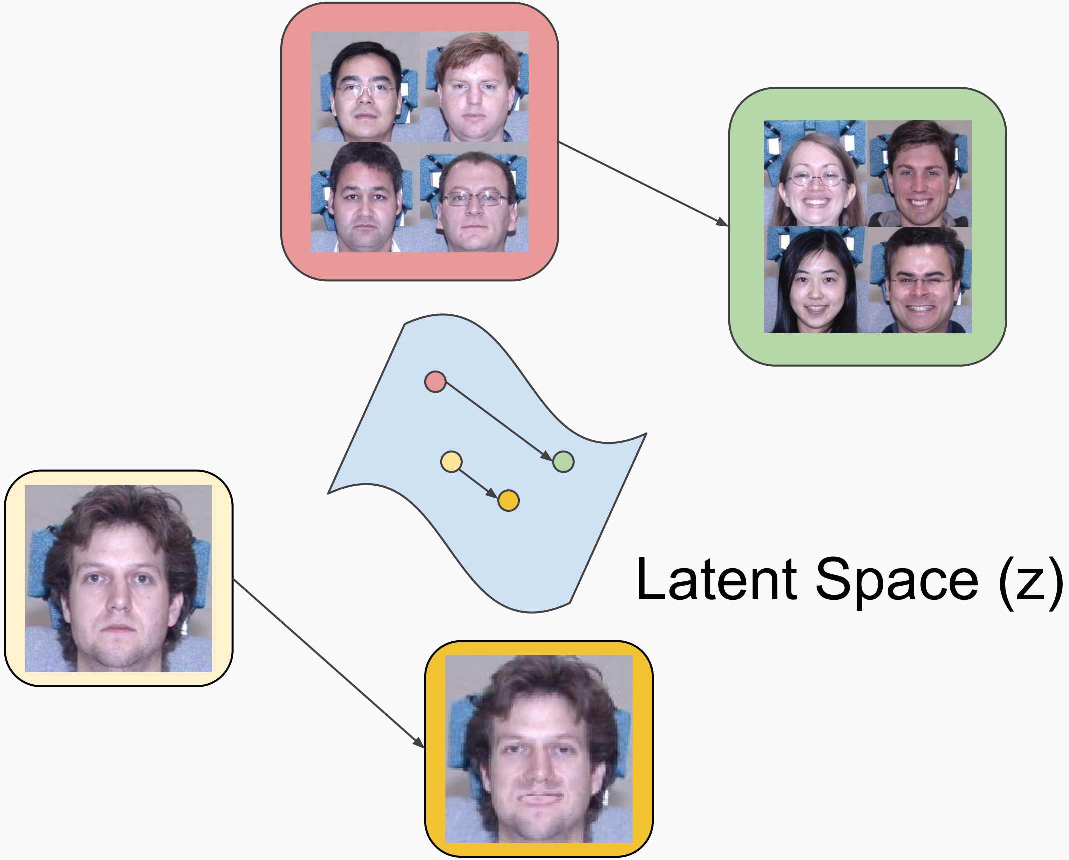


Image Credit: Semantic Facial Expression Editing Using Autoencoded Flow

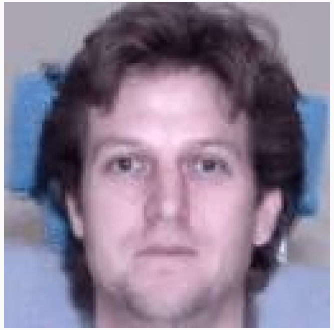
Results: Expression Editing

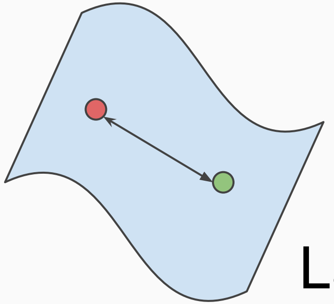
Suppress Original Magnify

Original Squint

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

Closing Remarks

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
- Xinchen Yan, Jimei Yang, Kihyuk Sohn, Honglak Lee, Attribute2Image: Conditional Image Generation from Visual Attributes, ECCV, 2016
- Jacob Walker, Carl Doersch, Abhinav Gupta, Martial Hebert, An Uncertain Future: Forecasting from Static Images using Variational Autoencoders, ECCV, 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
- Raymond Yeh, Ziwei Liu, Dan B Goldman, Aseem Agarwala, Semantic Facial Expression Editing using Autoencoded Flow, 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
- Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, Ole Winther, Auxiliary Deep Generative Models arXiv, 2016