Hyperparameter effects on learning

[Data](#_vvq0utwmhodm)

[Synthetic training data](#_h1ntjka8iszj)

[Synthetic testing data](#_m1o412bypnha)

[Adversarial testing data](#_y2e9u8nt1nxr)

[Real data](#_7mdmevo5zhhp)

[Model](#_4qjox26hcmve)

[2-layer Convolutional Variational Autoencoder (VAE) Architecture:](#_eohhawjzj7c2)

[Input pre-processing:](#_eu74o1yxkxme)

[Theoretical Loss function:](#_6e8mjrpcgib)

[Practical Loss function:](#_sur9aom3pa4f)

[Ways to diagnose if model accurately estimated posterior:](#_xq1t6gh8utds)

[Out-of-distribution detection:](#_xq1t6gh8utds)

[Extensions](#_6ch06xu1b9fj)

[Experiments](#_424lmslbdpmb)

[Experiment #1: Train only with strings containing A,T,G,C. Use 1 or 2 filters and latent dimensions.](#_mr6h21js0n1)

[Experiment #2: Train with strings containing A,T,G,C and A,U,G,C. Vary # of filters, latent dimensions, batch size, learning rate.](#_emysg6qcbmkm)

[Experiment #3: Repeat experiment #2 with more epochs to test whether KL loss converges](#_q5tsb1xkb10a)

## 

## 

# Data

## Synthetic training data

1. Generate 100,000 strings of length 576 such that there are alternating 3-mers and 6-mers generated from {A,**T**,G,C}.
2. Generate 100,000 strings of length 576 such that there are alternating 3-mers and 6-mers generated from {A,**U**,G,C}.

**Constraint:** In addition, ensure that the same 3-mers are placed in the same positions for all strings. The 6-mers may be random. In total, we have 200K reads with a vocabulary of either {A,T,G,C} or {A,U,G,C}.

## Synthetic testing data

1. Generate 100,000 strings of length 576 such that there are alternating 3-mers and 6-mers generated from {A,**T**,G,C}.
2. Generate 100,000 strings of length 576 such that there are alternating 3-mers and 6-mers generated from {A,**U**,G,C}.

**Constraint #1:** ensure that the same 3-mers are placed in the same positions for all strings. The 6-mers may be random.   
**Constraint #2:** The test strings are unique from the training strings

## Adversarial testing data

1. Generate 100K random strings of length 576

**Constraint #1:** Each string can only have a vocabulary of either {A,T,G,C} or {A,U,G,C}.

## Real data

1. Generate 100,000 strings of length 576 from the 337 viral strains (discovered pre-2018).

Constraint #1: Only sample strings from the first 1,000 base pairs of the viral genomes.

Constraint #2: No sequencing errors/noise are introduced in the creation of this dataset. I.e. ART is not used.

# 

# Model

## 2-layer Convolutional Variational Autoencoder (VAE) Architecture:

|  |  |
| --- | --- |
| **Encoder** | **Decoder** |
| Layer 1: Input layer | Layer 1: Fully-connected layer |
| Layer 2: Temporal convolutional layer (Convolves along the sequence dimension) | Layer 2: Transpose Convolutional layer |
| Layer 3: Fully-connected layer | Layer 3: Convolutional layer |
| Layer 4: Sampling layer | Layer 4: Output layer |

\*The number of filters, activations, hidden units, etc will be presented in the experiments section.

## **Input pre-processing:**

The input characters were alphabetically one-hot encoded such that (A,C,G,T,U) --> (00001, 00010, 00100, 01000, 10000)

## **Theoretical Loss function:**

**Notation**

We define as the input, as the latent (encoded) variable, and as the reconstructed input (the output). We refer to the learnable parameters of the encoder as . We refer to the learnable parameters of the decoder as .

**Terminology:**

* 1. We refer to as the marginal likelihood. It is also referred to as the *evidence.*
  2. We refer to as the prior and as the posterior
  3. The reconstruction is log likelihood

**Goal:**

If our main goal is compute an accurate low-dimensional representation of our data, then we focus on calculating . If we also care about the reconstruction, then we also try to calculate .

**Latent distribution**

We estimate the posterior using where represents a family of distributions (e.g. if were gaussian, then ). Assuming is a gaussian, then the sampling generates mean and log variance. If we want to model a different distribution, we simply redefine the sampling layer to generate the new distribution’s variables.

**Loss calculation**

We represent as the total loss and as the loss for input . The loss for is then the expected negative log likelihood of the reconstruction plus a regularizer (the KL divergence)

The reconstruction loss can be defined in many ways, specific to the problem. For example, in our case, the reconstruction for input is the sum of the cross entropy loss for each value in the string . We define reconstruction loss as .

The KL loss is defined as follows:

We see that the log model evidence is . It is an integral because we have to calculate the joint distribution for all configurations of z.

The final term is the ELBO and lower bounds the marginal log likelihood. We define

Combining the log model evidence and KL loss, we can rewrite it as:

In application, for numerical stability, we minimize the loss by minimizing negative .

## **Practical Loss function:**

We define the reconstruction loss of a single datapoint as the sum of the cross entropy () at each value of that datapoint:

where is the dimension of the .

We define the KL loss as :

where is the dimension of .

## **Ways to diagnose if model accurately estimated posterior:**

In practice, is usually a large vector. To check if model is theoretically able to capture the data distribution, define to be so large that many values in are , indicating an *over-complete* state. If a large is still not able to capture the data, then the problem is with KL divergence.

The reason we do this is because if we were to just view the KL loss, it might actually converge, but doesn’t mean it has truly converged at the possible minimum.

## **Out-of-distribution detection:**

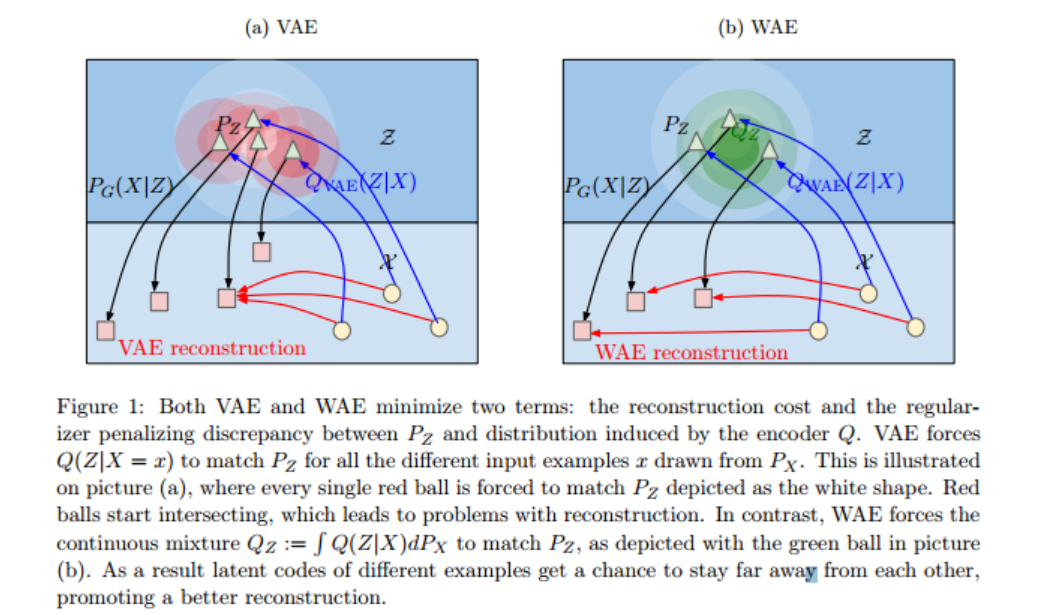
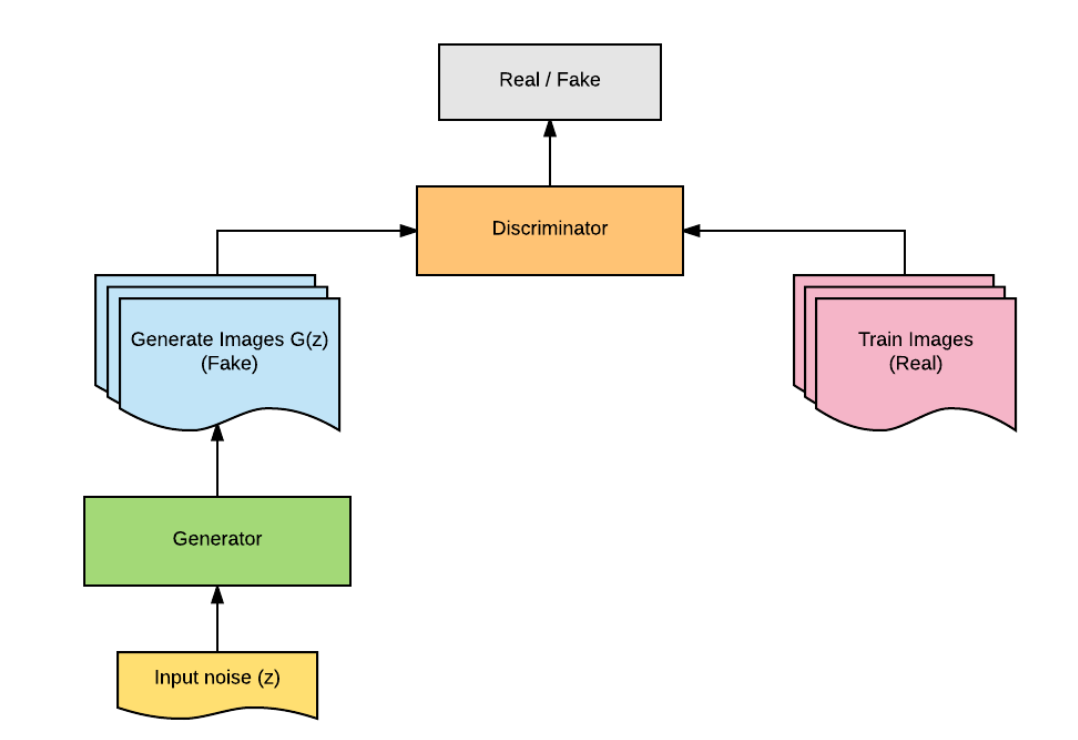
The reconstruction and KL loss should be low for in-distribution samples and high for out-of-distribution samples.

## **Extensions**

There exist many extensions to VAEs. Consider the following:

1. For very complex datasets, importance weighted autoencoders:

The variational autoencoder (VAE; Kingma, Welling (2014)) is a recently proposed generative model pairing a top-down generative network with a bottom-up recognition network which approximates posterior inference. It typically makes strong assumptions about posterior inference, for instance that the posterior distribution is approximately factorial, and that its parameters can be approximated with nonlinear regression from the observations. As we show empirically, the VAE objective can lead to overly simplified representations which fail to use the network's entire modeling capacity. We present the importance weighted autoencoder (IWAE), a generative model with the same architecture as the VAE, but which uses a strictly tighter log-likelihood lower bound derived from importance weighting. In the IWAE, the recognition network uses multiple samples to approximate the posterior, giving it increased flexibility to model complex posteriors which do not fit the VAE modeling assumptions. We show empirically that IWAEs learn richer latent space representations than VAEs, leading to improved test log-likelihood on density estimation benchmarks. <https://arxiv.org/abs/1509.00519>

1. Wasserstein autoencoder: 
2. VAEs are good at latent representations. But if our goal is good reconstruction, then we can use a generative adversarial network:  
   Essentially, the generator is improved by using the loss function of the discriminator. If the discriminator is easily able to detect the fake image generated by the generator, then the generator is doing a poor job, so it is updated more. But if the discriminator is doing a poor job, then the generator is doing great.

# 

# 

# Experiments

## Experiment #1: Train only with strings containing A,T,G,C. Use 1 or 2 filters/latent dimensions.

1. Did the loss converge?
2. Are the reconstructions unique for strings with only A,T,G,C?
3. Are the reconstructions unique for adversarial samples?

## Experiment #2: Train with strings containing A,T,G,C and A,U,G,C. Vary # of filters, latent dimensions, batch size, learning rate.

1. Did the loss converge?
2. Are the reconstructions unique?
3. What is the trade-off between accuracy and number of filters/latent dimensions?
4. What is the trade off between accuracy and batch size/learning rate?

## Experiment #3: Repeat experiment #2 with more epochs to test whether KL loss converges

1. Did the loss converge?
2. What is the roc curve for classification(based on loss threshold)?

Experiment #4: Use best model for experiment #3. Create new training data using real viral samples.