A Robust Hierarchical Variational Auto-encoder for Unsupervised Star Galaxy Classification

Author

Abstract

Keywords:

---

1. Introduction

A major advantage of deep learning is that useful features can be learned automatically from images. In Kim & Brunner's previous work, they showed that convolutional neural networks (ConvNets) are able to produce accurate star-galaxy classifications by learning directly from the pixel values of photometric images.

However, when using supervised learning algorithms, we need a lot of manual tagging data or images, which is pretty time/resource consuming. If we can use a unsupervised learning way to extract useful features automaticly and then run clustering algorithms on those extracted features, we may realize unsupervised star-galaxy classification as well as pixel wise segmentation.

Auto-Encoders(AEs) and Variational Auto-Encoders(VAEs) are powerful tools for unsupervised learning and feature extracting.

In our star-galaxy classification and segmentation task, the X represents $64 * 64 * 5$ input images to be classified, so that $\mathcal{X}$ has $64 * 64 * 5$ dimensions. Ideally, we only need one hidden variable to represent each image as a star or a galaxy, so that $\mathcal{H}$ have 1 demension. In order to get better visualization and end-to-end classification result, we choose to use both one and more hidden variables in practice. There are other ideas about the choice of dimensions of hidden variables, which will be explained later.

The aim of this work is to perform unsupervised pixel-wise star-galaxy classification and segmentation, which means we first segment the objects, including stars and galaxies from the background, and then classify them into the two classes, learning directly from the pixel values of photometric images.

In the following part of this article, we first introduce our dataset and some other preliminaries for AEs & VAEs in part 2; part 3 and part 4 will concentrate on the classification task and segmentation task separately, including several different methods as well as their pros and cons; the last two parts will sum up the results.

2. Data set

In this project, I used the SDSS data set. Taking into account the need for adequate training of the model as well as the computing power, I randomly picked 140,000 images and separate them into training set(100,000 images), validation set(20,000 images) and test set(20,000 images) with each image has the size of $64 \times 64 \times 5$ (here are 5 channels: u,g,r,i,z).

As a matter of routine, we converted the brightness unit into magnitude m instead of using the luminosity L directly, using

$$m_{obj} = m_{\odot} - 2.5 \log \left[\frac{L_{obj}}{L_{\odot}} \left(\frac{d_{\odot}}{d_{obj}} \right)^2 \right] \tag{1}$$

3. Preliminaries

3.1. Preliminaries for AEs/VAEs

In AEs, there are always two parts: the encoder and the decoder(as is shown in Fig.1), which can be defined as transitions ϕ and ψ, such that

$$\phi : \mathcal{X} \to \mathcal{H} \tag{2}$$

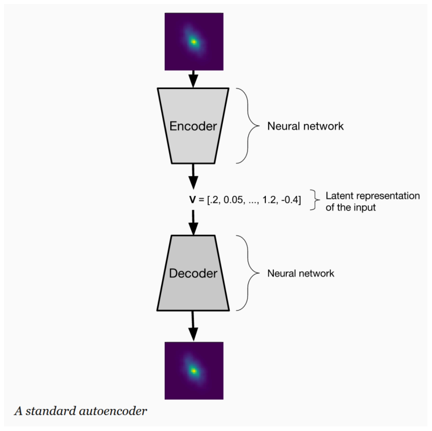
$$\psi : \mathcal{H} \to \mathcal{X} \tag{3}$$

$$\phi, \psi = \arg\min_{\phi,\psi} \parallel X - (\phi \circ \psi) X \parallel^2 \tag{4}$$

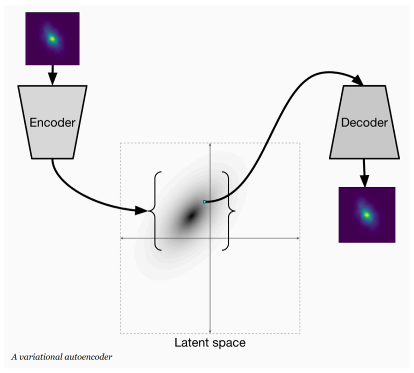
here the $\mathcal{X}$ denotes the high-dimensional input space where we sample our data X from. In most popular situations, X are images with thousands or millions of dimensions (pixels). The $\mathcal{H}$ here denotes the hidden space, which usually has lower demensions.

The structure of VAEs is similar to AEs though the mathematical intuition behind AEs and VAEs are quite different. An easier way to have a general concepts of VAEs is to use AEs as an analogy.

In AEs, we map the input (images) space $\mathcal{X}$ to a low dimensional hidden space $\mathcal{H}$ (See Figure 1(a)). The loss function of AE measures the pixel-wise difference between input images and output images. Thus we may call this self-supervised learning. By performing down-sample and upsample process, most hot pixels can be removed.


Encoder
Neural network
V = [.2, 0.05, ..., 1.2, -0.4]
Latent representation of the input
Decoder
Neural network
A standard autoencoder

(a) AE



(b) VAE

Figure 1: structure of AE/VAE

However, if we want to use some certain $V \in \mathcal{H}$ to generate a image, the result may be inconsistent. Because the $\mathcal{H}$ space is not that smooth. Then here comes concepts of the variational parts. VAEs are often used as a generative models to generate new samples by a probability density function(PDF) $P(X)$ that comes from the same distribution $P_{gt}(X)$ as training data $P_{data}(X)$ defined over $\mathcal{X}$.

For the Encoder part of the VAEs, a certain class of input images are mapped to a certain Multi-dimensional, depending on the number of hidden variables, Gaussian distribution, as is shown in Figure 1(b). And then the Decoder uses a resampled hidden value to generate a new image.

Different images will be mapped to different Gaussian envelope in $\mathcal{H}$. And the images looks alike should be mapped to neighborhood in $\mathcal{H}$. As a result, the hidden space $\mathcal{H}$ will consist of thousands or millions of Gaussians, as is shown in Figure 2. Notice that any distribution in d dimensions can be generated by taking a set of d variables that are normally distributed and mapping them through a sufficiently complicated function, a traditional assumption is that $V \sim \mathcal{N}(0, I)$, where I is the identity matrix. This assumption work

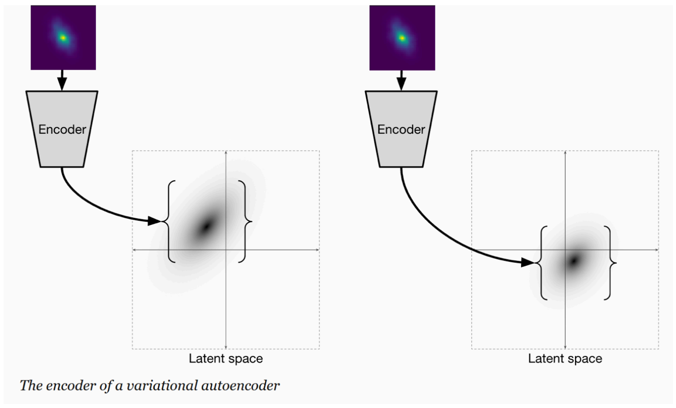


Figure 2: Encoder of VAE

as a constrain in the hidden space by adding the KL-divergence term,

$$D_{KL}(P,Q) = \int_{-\infty}^{\infty} P(x) \log \frac{P(x)}{Q(x)} dx \tag{5}$$

which discribe the logarithmic difference between the probabilities P and Q, to the loss function:

$$L_{VAE} = \parallel X_{in} - X_{out} \parallel^2 + D_{KL}(P(V), \mathcal{N}(0,I)) \tag{6}$$

where

$$P(V) \sim \sum_{i=1}^{M} \mathcal{N}(\mu_i, \sigma_i^2) \tag{7}$$

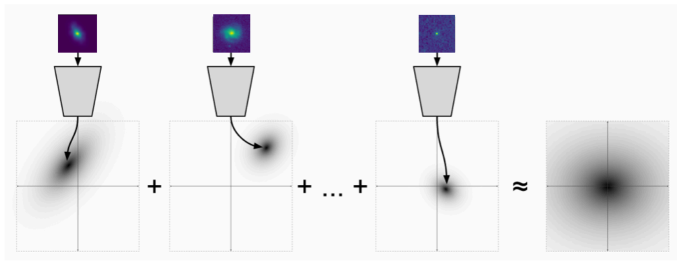
denotes the distribution sum over M training samples and μ_i, σ_i denote for the mean and standard deviation of each Gaussian in the hidden space $\mathcal{H}$.

3.2. Revise of the KL-term

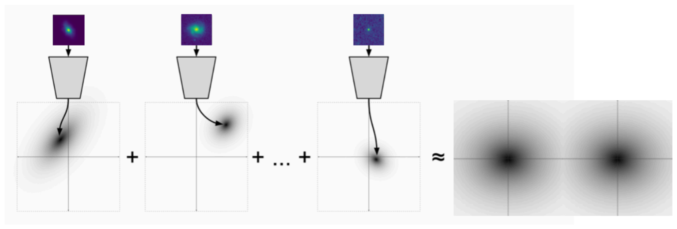
Having this Gaussian prior term, we can use any point from the Gaussian in this hidden space to generate a quite consistent image that at least looks like one of input samples of training set. This $\mathcal{N}(0,1)$ prior works well in generating new images, but not conducive to unsupervised classification. To separate different classes into two or more clusters, a better choice of the priori distribution should be a double-peak Gaussian.

$$Q(V) \sim \frac{1}{2}N(-m, s^2) + \frac{1}{2}N(m, s^2) \tag{8}$$

Figure 3(a) shows the optimization goal when we use $\mathcal{N}(0, I)$ prior and Figure 3(b) shows the optimization goal when we use double-peak-Gaussian prior.



(a) prior $= \mathcal{N}(0, I)$



(b) prior $= \frac{1}{2}N(-m, s^2) + \frac{1}{2}N(m, s^2)$

Figure 3: Hidden distribution with different priors

Calculation of the KL-divergence of multi-peak-Gaussian case calls for lots of work. Here we propose an approximation of scaling method, which is an easier approach. And a exact calculation is also included in Appendix A.

Sometimes when we train the neural net with a KL-divergence or JS-divergence, we meet the gradient disappearence problem. And this is exactly the problem people met when training Generative Adversarial Networks(GANs). We also proposed two analogies for Wasserstein metric, called AW and PW. For more intuitions and interpretations as well as calculation of this part, please refer to Appendix B.

4. Star-Galaxy Classification

4.1. Methods

In the classification task, we hope we can make use of VAEs' dimensionality reduction ability to make clustering easier. And we try to use different clustering algorithms to separate the hidden variables of in the hidden space. However, VAEs are not designed to be a classifier but a generator. In order to get better classification performance, we tried different approachs: we first tried to use traditional VAEs and Manifold learning algorithms and set the result as a baseline; then we tried to revise the loss function of VAEs as proposed above; finally we tried to use a hierarchical structure which uses an AE to extract low-level features and uses a VAE to do classification. The intuitions for each method are as depicted below.

4.1.1. VAE + Manifold learning(V+M)

The first idea is to use the commonly used method of unsupervised clustering. This method has three steps. First, use VAEs or AEs to extract features from original images and reduce the dimensions into d_{hid}, where $d_{hid} \ll D_{input}$. Then, run a manifold learning algorithm(such as ISOMAP, t-SNE etc.) to map the d dimensional d_{hid} into a 1 dimensional scalar. For the last step, use the scalar and a threshold(can be determined by prior knowledge like star-galaxy proportion) to perform classification.

4.1.2. VAE with revised KL-term + Manifold learning(Vr+M)

As the previous analysis has shown in 2.3, we use a VAE with revised KL-term here instead of using the traditional KL-term directly like first step in 3.1.1.

We tried to use 4 different kinds of revised KL-term(including the analogy terms of Wasserstein Metrics) instead of the traditional one, details of which are shown in Appendix A. Those loss terms are double-peak KL-Scaling(SC), double-peak KL(DKL), analogy of Wasserstein loss(AW) and pseudo Wasserstein loss(PW).

4.1.3. AE + VAE with revised KL-term(A+Vr)

The most successful generative models often use only a single layer of latent variables [1], and those that use multiple layers only show modest performance increases in quantitative metrics such as log-likelihood [2].

When using more hidden variables (e.g. 30), the representation ability would be more sufficient to generate consistant images. But the features are

[1]Radford et al., 2015; van den Oord et al., 2016
[2]Snderby et al., 2016; Bachman, 2016

more likely to be in lower-level. So that unsupervised clustering methods can not work here.

When using less hidden variables (e.g. 2), the representation ability would be less sufficient to generate consistant images, although the features are more likely to be in higher-level. So that accuracy of unsupervised learning is limited.

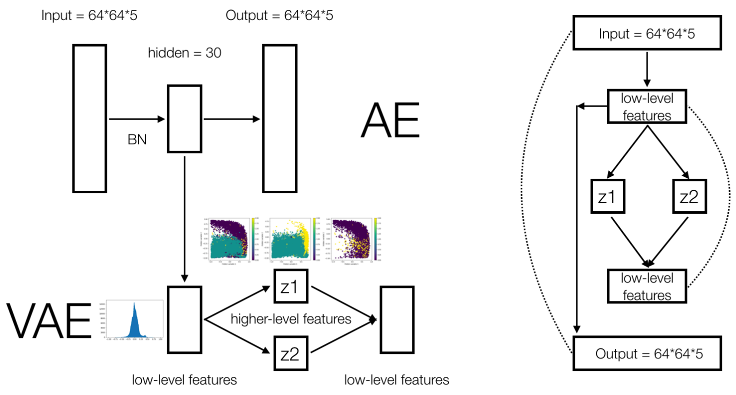


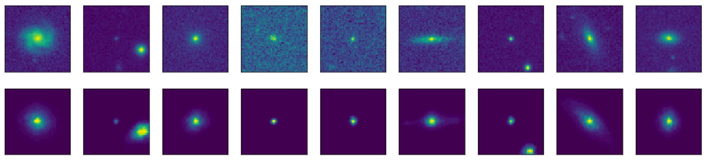
Figure 4: Left:AE + VAE; Right: VAE with the loss defined by AE

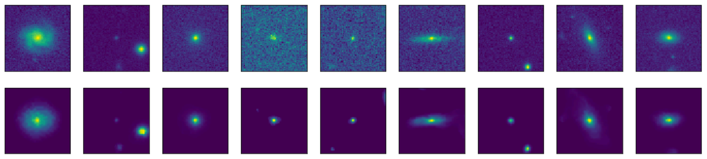
To combine the pros and avoid the cons mentioned above, we proposed a hierarchical structure, which can be understand as a VAE with the loss function defined by an AE(See Figure 4(b)). From another perspective, the AE here, which has 30 hidden variables, act as an low-level feature extractor, while the VAE here act as an high-level classifier. Such structure has several advantages: First, reproduce the original images with 30 hidden variables is pretty easy. To prove this conclusion, we tried to use a supervised learning method whose inputs are those 30 hidden variables. And an accuracy of 93% was obtained. Second, using Batch Normalizaition method, the final classification result becomes less sensitive to the normalization method used for the input data, thus the model become more robust. Therefore, there will be less hyper-parameters to tune. Third, we can then use a VAE with a multi-peak Gaussian prior to perform better classification. Last but not least, the optimization goal for the VAE here is to reproduce the 30 hidden variables of the AE instead of reproduce the 64*64*5 dimensional images, which is much easier to access.

5. Segmentation

Here we used Hypercolumns (by Hariharan et al.) to combine the features extracted by AEs/VAEs.

In the segmentation task, the variational part in VAEs can not help to improve the neural net's performance, for we don't aim at using any point in the hidden space to generate images that look real. To obtain better performance, I tried to use residual connections between each down-sampling layer and the correspond up-sampling layer.


(a) VAEs + Hypercolumns


(b) AEs + Hypercolumns

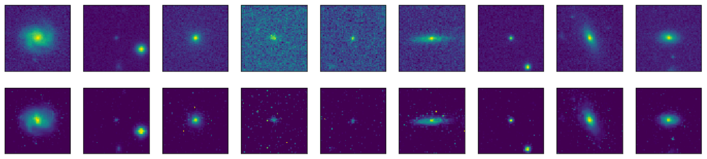

(c) AEs + Residual connections + Hypercolumns

Figure 5: Hidden distribution with different priors

5.1. Method

5.1.1. VAEs + Hypercolumns

When using VAE(Figure 5(a)), there are some shape distortion and blurred edge. Those are not conducive to pixel-level segmentation.

5.1.2. AEs + Hypercolumns

If we use AE instead of VAE(Figure 5(b)), most stars and galaxies can be found. But some of the very faint objects are missed.

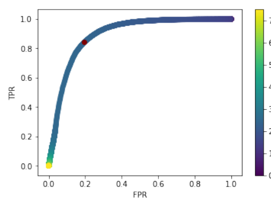
5.1.3. AEs + Residual connections + Hypercolumns

If we use AEs with residual connections(Figure 5(c)), then more structure informations can be kept in the output layer. In the first image of the Figure 5(c), the spiral structure is also reproduced.

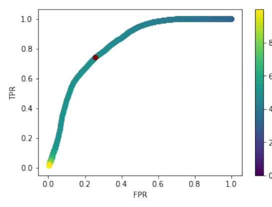
However, more noises are also included in output images as a cost. In practice, we may change the layers we choose to use and turn their weights as a trade-off.

6. Experiments

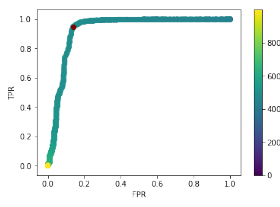
6.1. Classification



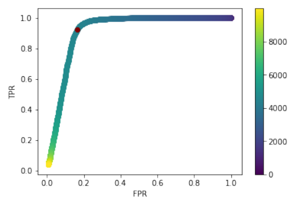
(a) V+M: AUC=0.89



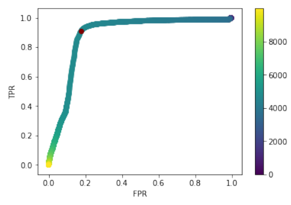
(b) Vr(SC)+M: AUC=0.82



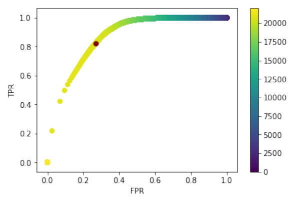
(c) Vr(AW)+M: AUC=0.92



(d) Vr(DKL)+M:AUC=0.91



(e) Vr(PW)+M: AUC=0.88



(f) A+Vr(KL): AUC=0.89

Figure 6: ROC and AUC of each method

Figure 6 shows some of the result we get using the methods in Part 3. The first 5 classification results are quite sensitive to the initializers of neural network. Actually such instability comes from the manifold learning algorithm we use after VAEs. Manifold learning algorithms are pretty sensitive to the outliers. However, there are lots of multi-object images in our dataset, which means sometimes stars and galaxies may appear in a certain image. But the label of each image comes from the object in the center.

6.1.1. VAE + Manifold learning(V+M)

TODOs:
Repeat
Stats?

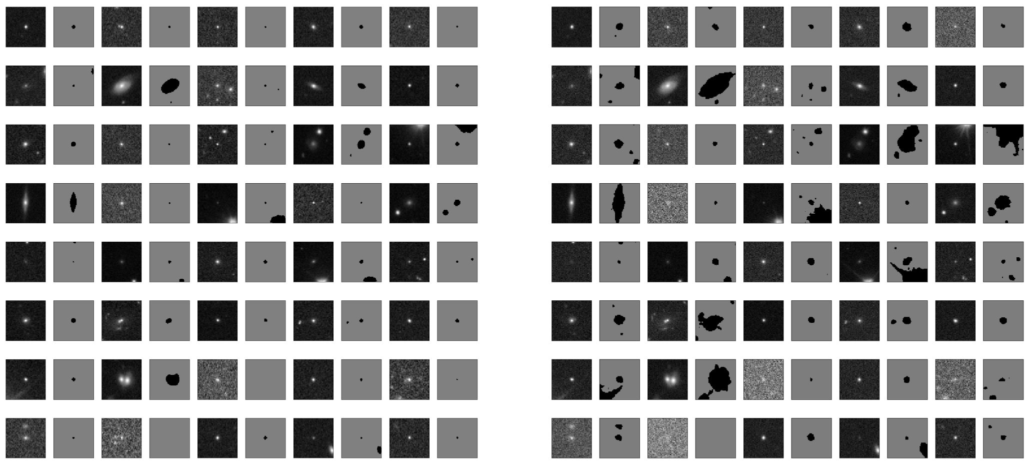
6.1.2. VAE with revised KL-term + Manifold learning(Vr+M)

TODOs:
Repeat
Stats?

6.1.3. AE + VAE with revised KL-term(A+Vr)

TODOs:
Repeat
Stats?

6.2. Segmentation



(a) AE + Hypercolumns (b) AE + residual connection + Hypercolumns

Figure 7: Segmentation result

Figure 7(a) shows the result of segmentation using AE and Hypercolumns. Figure 7(b) shows the result of segmentation using AE with residual connections and Hypercolumns. More detail informations in the original images are included in Figure 7(b). However, more noise pixels are recognized as objects and sometimes several objects are segmented as one.

The method using AE + residual connection + Hypercolumns will definitly be more sensitive to faint stars, while the result become more precise without the residual connections.

6.3. *Segmentation+Classification*

Figure 8 is a demo of the combination of the Segmentation net and the Classification net. For each image in Figure 8, gray pixels are recognized as backgroud pixels, red pixels are recognized as stars/QSOs while orange pixels are recognized as galaxies. The groundtruth labels are above each figure.

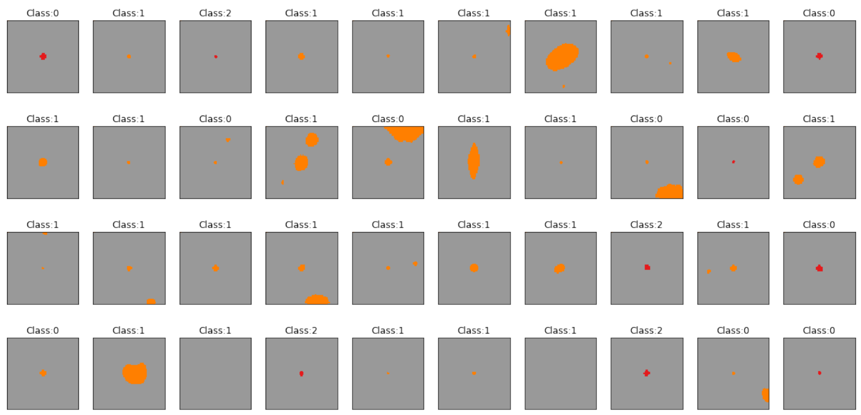


Figure 8: Segmentation result

7. Conclusion

TODOs: Conclusion

Appendix A. The calculation of $D_{KL}(P, Q)$

Appendix A.1. An scaling technique

$$
\begin{aligned}
& D_{KL}\left(N(\mu,\sigma^2)\| \frac{1}{2}N(-m,s^2)+\frac{1}{2}N(m,s^2)\right) \\
=& D_{KL}\left(2*\frac{1}{2}N(\mu,\sigma^2)\| \frac{1}{2}N(-m,s^2)+\frac{1}{2}N(m,s^2)\right) \\
\leq& D_{KL}\left(\frac{1}{2}N(\mu,\sigma^2)\| \frac{1}{2}N(-m,s^2)\right)+D_{KL}\left(\frac{1}{2}N(\mu,\sigma^2)\| \frac{1}{2}N(m,s^2)\right) \\
=& -\frac{1}{2}\log 2\left(s^2+\log\sigma^2-\frac{1}{2}(\mu-m)^2-\frac{1}{2}(\mu+m)^2-\sigma^2\right)
\end{aligned}
$$

Appendix A.2. Exact solution

$$
\begin{aligned}
&D_{KL}\left(N(\mu,\sigma^2)\|\frac{1}{2}N(-m,s^2)+\frac{1}{2}N(m,s^2)\right)\\
=&\int_{-\infty}^{\infty} N(\mu,\sigma^2)\log\frac{N(\mu,\sigma^2)}{\frac{1}{2}N(-m,s^2)+\frac{1}{2}N(m,s^2)}dx\\
=&\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}\log\frac{\frac{1}{\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\frac{1}{2s}[e^{-\frac{(x-m)^2}{2s^2}}+e^{-\frac{(x+m)^2}{2s^2}}]}dx\\
=&\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}\log\frac{2s}{\sigma}dx\\
-&\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}[\frac{(x-\mu)^2}{2\sigma^2}-\frac{(x-m)^2}{2s^2}]dx\\
-&\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}\log[1+e^{-\frac{2mx}{s^2}}]dx\\
=&\alpha-\beta-\gamma
\end{aligned}
$$

$$
\begin{aligned}
\alpha &= \log\frac{2s}{\sigma}\\
\beta &= -\frac{(m-\mu)^2+\sigma^2-s^2}{2s^2}\\
\gamma &\approx -\frac{2m[-\sigma e^{-\frac{\mu^2}{2\sigma^2}}+\sqrt{\frac{\pi}{2}}\mu Erfc(\frac{\mu}{\sqrt{2}\sigma})]}{s^2}
\end{aligned}
$$

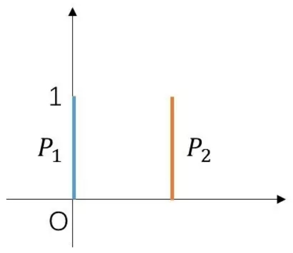
with an approximation of $Erfc(x)\approx 1-tanh(1.19x)$

$$
\begin{aligned}
&D_{KL}\left(N(\mu,\sigma^2)\|\frac{1}{2}N(-m,s^2)+\frac{1}{2}N(m,s^2)\right)\\
\approx&\log\frac{2s}{\sigma}+\frac{(m-\mu)^2+\sigma^2-s^2}{2s^2}\\
+&\frac{2m\{-\sigma e^{-\frac{\mu^2}{2\sigma^2}}+\sqrt{\frac{\pi}{2}}\mu[1-tanh(1.19\frac{\mu}{\sqrt{2}\sigma})]\}}{s^2}
\end{aligned}
$$

Appendix B. Defect of KL-divergence and analogies of Wasserstein metric

Appendix B.1. Defect of KL-divergence

Sometimes when we train the neural net with a KL-divergence or JS-divergence, we meet the gradient disappearence problem: in the figure below, P_1 and P_2 are two dirac δ functions (we may suppose they are two gaussian distributions with almost no overlap)



The KL divergence of P_1 & P_2 is:

$$D_{KL}(P_1, P_2) = \begin{cases} \infty & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

The JS divergence of P_1 & P_2 is:

$$f(x) = \begin{cases} log2 & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

Most of the time the derivatives of KL-divergence and JS-divergence are zero.

But if we use Wasserstein metric here,

$$Wassersteinloss = ||\theta||_p$$

which is continuous. This is quite like what Arjovsky et al. did in Wasserstein GAN.

Appendix B.2. A naive analogy(AW)

With

$$P(x) \sim \sum_{i}^{m} N(\mu_i, \sigma_i^2) \quad Q(x) \sim \frac{1}{2}N(-m, s^2) + \frac{1}{2}N(m, s^2)$$

an naive way to avoid gradient disappearance is to introduce an analogy of Wasserstein loss (AW loss):

$$AW(P, Q) = -||\mu||_p + ||\sigma - s||_p$$

which makes μ as large as possible and constrain σ by the second term. In practice, there should be an tanh activation function in the last layer of the encoder (so that the value of μ is limited).

Appendix B.3. Another analogy(PW)

The defination of p^{th} Wasserstein metric is

$$W_p(\mu, \nu) := \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{M \times M} d(x, y)^p d\gamma(x, y) \right)^{1/p}$$

.

Another name of Wasserstein metric is Earth Mover Distance,which is pretty vivid. The Earth Mover Distance is talking about an analogy that we may understand the Wasserstein metric in this way: If we want to move one of a sand dune to another place, then the "W distance" is defined by the smallest work an earth mover need to do.

Back to our problem, here we need to calculate the optimal route of transportation, which is an combinatorial optimization problem. To avoid this difficulty, I tried to use another analogy. I called it Pseudo Wasserstein(PW) loss: With

$$P(x) \sim \sum_{i}^{m} N(\mu_i, \sigma_i^2) \quad Q(x) \sim \frac{1}{2}N(-m, s^2) + \frac{1}{2}N(m, s^2)$$

$$PW(P, Q) = ||\mu||_p + \left(\int_{-\infty}^{\infty} ||P - Q||_p dx \right)^{1/p}$$

By this definition,

$$PW\left(N(\mu,\sigma^2)\|\frac{1}{2}N(-m,s^2)+\frac{1}{2}N(m,s^2)\right)$$

$$=||\mu||_p+\left(\int_{-\infty}^{\infty}||N(\mu,\sigma^2)-[\frac{1}{2}N(-m,s^2)+\frac{1}{2}N(m,s^2)]||_p dx\right)^{1/p}$$

$$=\frac{1}{4\sqrt{\pi}s\sigma\sqrt{\frac{1}{s^2}+\frac{1}{\sigma^2}}}\exp\left(-\frac{m^2}{s^2}-\frac{\mu^2}{2(s^2+\sigma^2)}\right)*\alpha$$

$$\alpha=-2\sqrt{2}\exp\left(\frac{m[-2\mu s^2+m(s^2+2\sigma^2)]}{2s^2(s^2+\sigma^2)}\right)-2\sqrt{2}\exp\left(\frac{m[2\mu s^2+m(s^2+2\sigma^2)]}{2s^2(s^2+\sigma^2)}\right)$$

$$+\sigma\sqrt{\frac{1}{s^2}+\frac{1}{\sigma^2}}\exp\left(\frac{\mu^2}{2(s^2+\sigma^2)}\right)+\sigma s(\frac{1}{s}+\frac{2}{\sigma}\sqrt{\frac{1}{s^2}+\frac{1}{\sigma^2}})\exp\frac{m^2}{s^2}+\frac{\mu^2}{2(s^2+\sigma^2)}$$

Appendix C. Model architecture graph

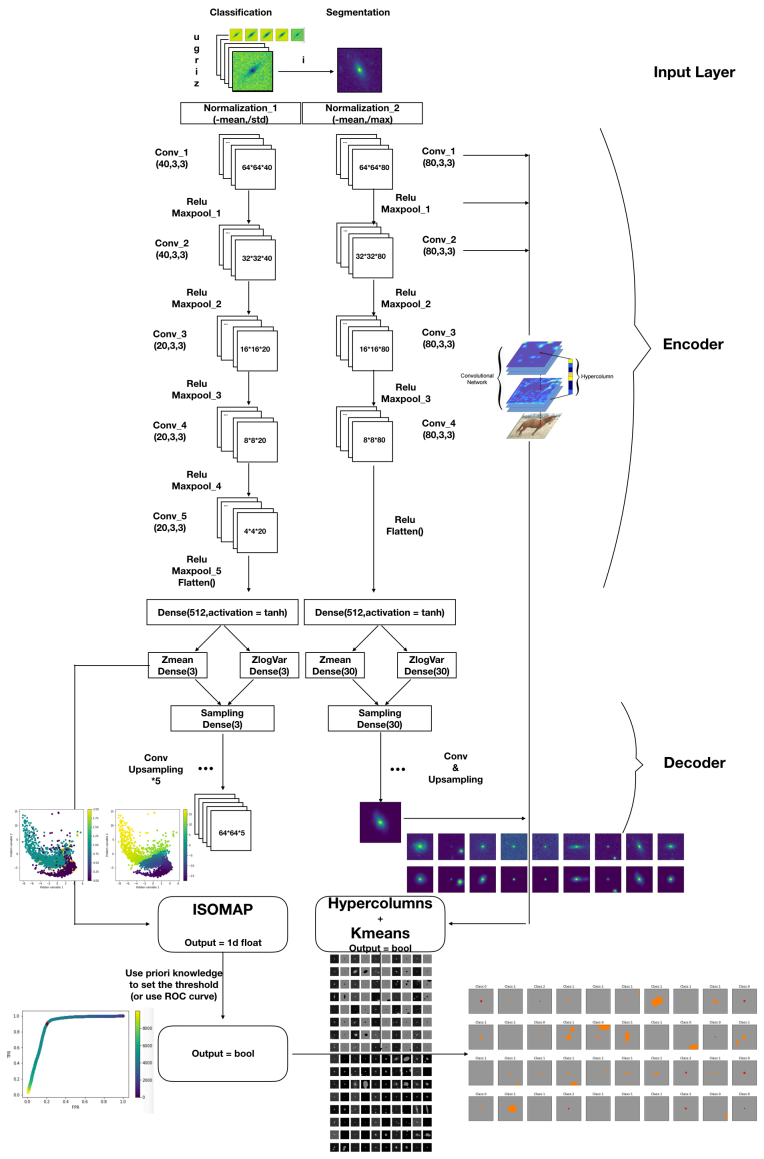


Figure C.9: Model architecture graph