

Project Report

An Expose on Kullback-Leibler Divergence

by

Faizan Shaikh Abdul Khalil Shaikh

for the course - UAI/500 Information Theory

Abstract

The writeup is an in-depth exploration of Kullback Leibler divergence (KLD), a widely used metric to measure two probability distributions. We will gently introduce the topic along with an example to familiarize ourselves with the terminologies. We will then dive into the mathematical aspects of KLD by formulating a theoretical derivation of KLD for two univariate gaussian distributions. Lastly, we will lay out a practical application of KLD as a learning metric for a deep learning architecture called Variational Autoencoders.

Introduction

One of my excursions into the Deep Learning space, surveying unsupervised deep learning, brought me to the topic of KL Divergence. Although I have encountered it once or twice, I have never explored the topic thoroughly. I figured this would be the opportunity to do so, as I can make this my class project for Information Theory (two for one deal :))

We've established that we are going to examine the term KL Divergence. But what exactly is KL Divergence? A textbook definition is that

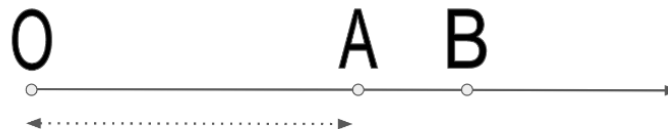
“Kullback-Leibler Divergence is a measure of comparison between two probability distributions”

Let's dissect this definition.

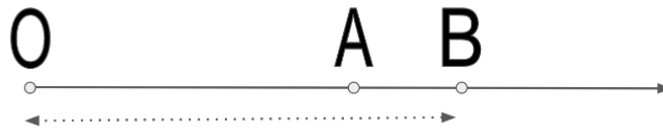
Say you have two points A and B on a line. How do you compare them?



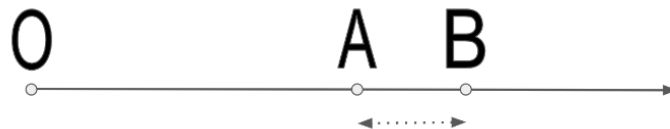
One way could be that we define an arbitrary point as origin O. Now we can calculate the distance between the origin and point A.



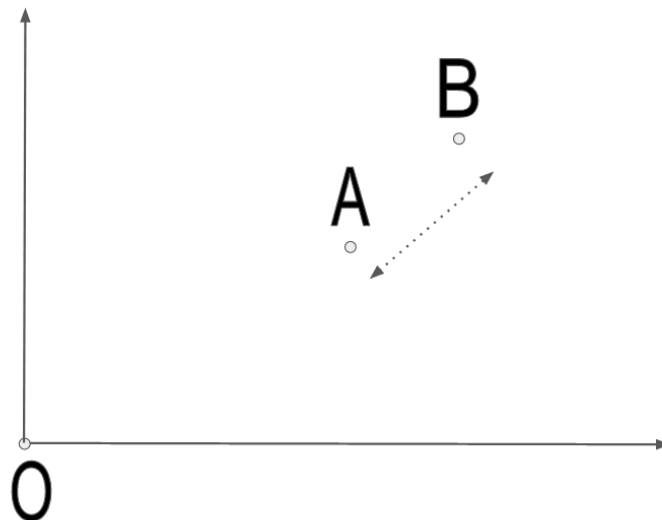
This comes out to be 3 units. In the same way, we can calculate the distance between the origin and point B



- which comes out to be 4 units. Now using these two quantities, we can calculate the distance between points A and B, which is 1 unit.

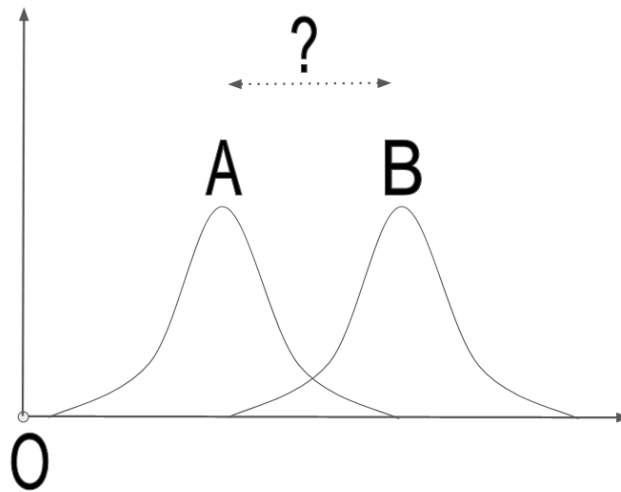


Simple enough? Lets extend this concept to a 2D plane



You can see that distance is still relevant as a way to compare two points.

But can we use the same way of comparison on two probability distributions?



As you might have guessed, we need a more elaborate way to first describe, and then compare, the two probability distributions. This is where KL Divergence comes in, where **we can calculate how much information needs to be changed to obtain the first distribution from the second**. Mathematically, KL Divergence can be defined as [1]

$$\begin{aligned} \text{KL}(p\|q) &= - \int p(\mathbf{x}) \ln q(\mathbf{x}) \, d\mathbf{x} - \left(- \int p(\mathbf{x}) \ln p(\mathbf{x}) \, d\mathbf{x} \right) \\ &= - \int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} \, d\mathbf{x}. \end{aligned}$$


where, $p(\mathbf{x})$ is usually the true or original distribution and $q(\mathbf{x})$ is the model distribution which is getting compared to $p(\mathbf{x})$. We will come back to the maths later, but first, let's understand KL Divergence with an example.

Example: Video Compression

To understand the nitty gritty of the concept, let's take an example of a data compression algorithm. Suppose you want to send hours of video content from one place to another. What would be the simplest way to send this massive data through the internet?

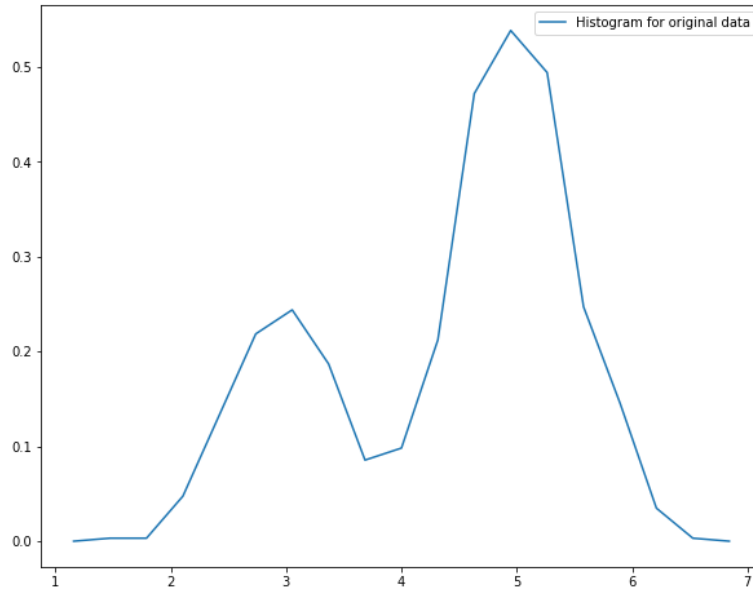


If you guessed it right, yes - we can reduce the resolution of the video (say from 1080p to 480p). So although the size of the data could become bearable, the compression would be done at the expense of the quality of the video. Then the question arises, how can we maintain the quality of the video to some extent but reduce the quantity substantially?

For our hypothetical scenario, let consider that a group of researchers came up with a state of the art Deep Learning architecture that does the job for us. But to accurately measure how good the algorithm does in comparison to the conventional method, we can use ( SOUND EFFECT DRUM ROLL) KL Divergence.

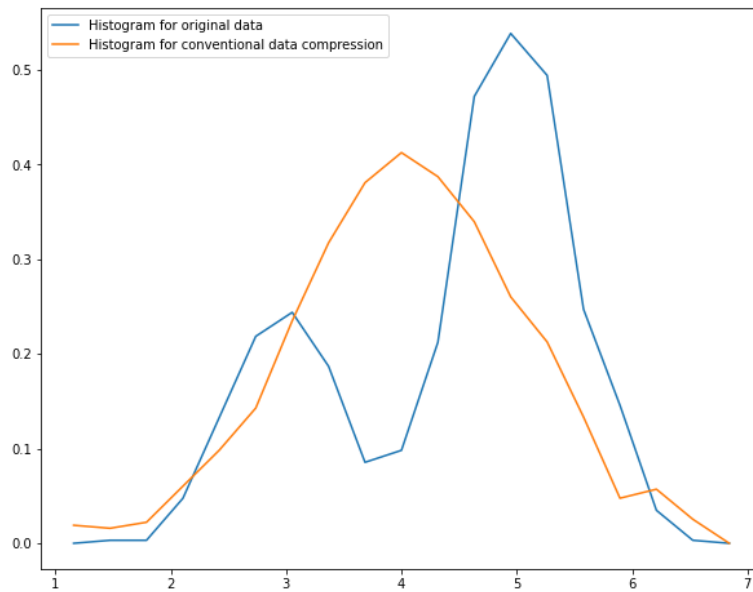
Lets formally lay down our requirement, we want to measure the performance of the SOTA DL architecture to the conventional method. We do this by computing how much information gets lost in both the methods. Ideally, we want the DL algorithm to have lower information loss during data compression.

For simplicity, we firstly do not consider the ordering of the data, but instead compare the contents and also, assign dummy values to the data. If we plot the histogram for our original video, it looks something like this



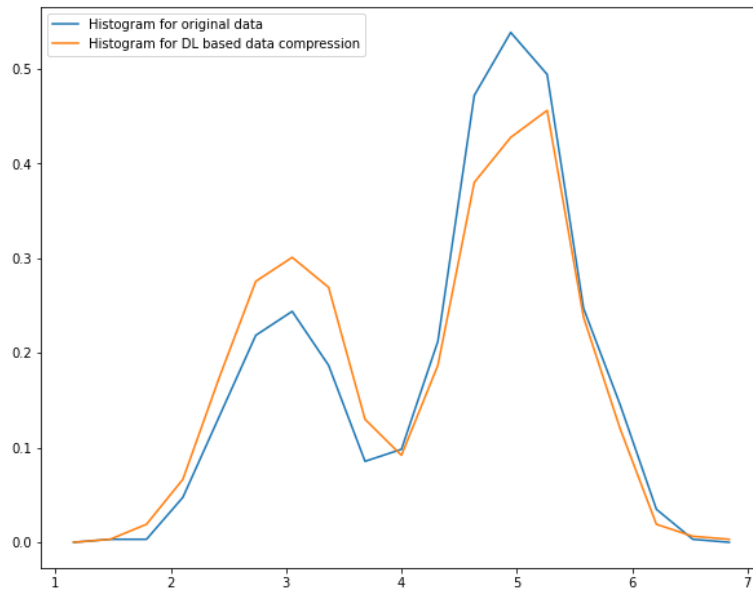
Technically speaking, this is a bimodal distribution with one smaller peak at 3 and another larger peak at 5.

Similarly, we can display the histogram formed by our conventional data compression technique, alongside the histogram of the original data



We see that instead of capturing the bimodal distribution, the technique tried to smooth out the data, giving us a generic unimodal gaussian distribution.

On the other hand, if we do the same thing for our DL technique, we get an visualization such as this



Although the DL technique isn't perfect, it still is able to capture the essence of the data. So just on the basis of these visualizations, we can conclude that the DL technique is better. But can we prove this with a measurable quantity?

This is where KL Divergence can help. If we numerically calculate the KL Divergence between the original video and the data compressed through conventional technique, it comes out to be 0.88. Whereas the KL Divergence between the original video and the data compressed through DL technique comes out to be 0.09 [3].

Therefore we can safely say that there is low information loss through DL based compression. All hail the DL gods!

Mathematical Representation of KL Divergence

In this section, in order to demystify the complex concepts, we will take a Q&A approach, try to get the overall understanding bit by bit

1. Can you explain to me in simple terms what KLD is?

I explained this in the previous section. But a TLDR is that KL Divergence is a measure of comparison between two distributions wherein we calculate how much information needs to be changed to obtain the first distribution from the second. It was first introduced in the paper “On Information and Sufficiency” [10] by S. Kullback and R. A. Leibler back in 1951.

2. What is the mathematical formula for KLD?

The formula for KL Divergence is as follows

$$\begin{aligned} \text{KL}(p||q) &= - \int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \left(- \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \right) \\ &= - \int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x}. \end{aligned} \quad [1]$$

where, $p(\mathbf{x})$ is usually the true or original distribution and $q(\mathbf{x})$ is the model distribution which is getting compared to $p(\mathbf{x})$. Both $p(\mathbf{x})$ and $q(\mathbf{x})$ are considered to be continuous distributions

3. Could you decompose the formula so that I can understand it better?

Consider the formula -

$$\text{KL}(p||q) = - \int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \left(- \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \right)$$

If you are in ML space, you might easily recognize that the first term is cross entropy, which is frequently used as a loss function for training supervised deep learning algorithms. And the second term is just the entropy of $p(\mathbf{x})$. Therefore

$$\begin{aligned} D_{\text{KL}}(P \parallel Q) &= \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{q(x)} - \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)} \\ &= H(P, Q) - H(P) \end{aligned} \quad [2]$$

where, $H(P, Q)$ is the cross entropy of P and Q and $H(P)$ is the entropy of P

4. The formula mentioned before is for continuous probability distribution functions. Can it be applied to discrete probability distributions?

Yes absolutely! Instead of integrating over a probability space, we consider the same formula with respect to the discrete possible points of x

$$D_{\text{KL}}(P \parallel Q) = - \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{Q(x)}{P(x)} \right) \quad [2]$$

5. What is the unit for measurement of KL Divergence?

If the logarithm has base e , the information is measured in nats. Otherwise if the logarithm has base 2, the information is measured in bits.

6. I understand what KLD is. Can you show me an example of how to calculate KLD for two gaussian distributions?

$$\begin{aligned} KL(P||Q) &= - \int p(x) \log(q(x)) - (- \int p(x) \log(p(x))) \\ &= \int p(x) \log(p(x)) - \int p(x) \log(q(x)) \end{aligned}$$

from the definition of gaussian distribution, we know that pdf of a gaussian is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Let's calculate $\log(p(x))$

$$p(x) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{\frac{-(x-\mu_1)^2}{2\sigma_1^2}}$$

$$\log(p(x)) = \log\left(\frac{1}{\sigma_1}\right) + \log\left(\frac{1}{\sqrt{2\pi}}\right) + \left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right)$$

$$\log(p(x)) = -\log(\sigma_1) + \log\left(\frac{1}{\sqrt{2\pi}}\right) - \frac{(x-\mu_1)^2}{2\sigma_1^2}$$

Similarly for $q(x)$

$$q(x) = \frac{1}{\sigma_2\sqrt{2\pi}} e^{\frac{-(x-\mu_2)^2}{2\sigma_2^2}}$$

$$\log(q(x)) = -\log(\sigma_2) + \log\left(\frac{1}{\sqrt{2\pi}}\right) - \frac{(x-\mu_2)^2}{2\sigma_2^2}$$

Replacing these values in formula,

$$KL(P||Q) = \int p(x) \log(p(x)) - \int p(x) \log(q(x))$$

$$= \int p(x) * (-\log(\sigma_1) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{(x - \mu_1)^2}{2\sigma_1^2})) - \int p(x) * (-\log(\sigma_2) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{(x - \mu_2)^2}{2\sigma_2^2}))$$

Because we integrating the equation over x, we can use these formulas

1. $\int N(x|\mu, \sigma^2)dx = 1$
2. $\int x * N(x|\mu, \sigma^2)dx = \mu$
3. $\int x^2 * N(x|\mu, \sigma^2)dx = \mu^2 + \sigma^2$

Lets focus on the first term

$$\begin{aligned} & \int p(x) * (-\log(\sigma_1) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{(x - \mu_1)^2}{2\sigma_1^2})) \\ &= \int p(x) * (-\log(\sigma_1) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{(x^2 + \mu_1^2 - 2x\mu_1)}{2\sigma_1^2})) \\ &= (-\log(\sigma_1) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{(\sigma_1^2 + \mu_1^2 + \mu_1^2 - 2\mu_1^2)}{2\sigma_1^2})) \\ &= (-\log(\sigma_1) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{1}{2})) \end{aligned}$$

Now let's focus on the second term

$$\begin{aligned} & \int p(x) * (-\log(\sigma_2) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{(x - \mu_2)^2}{2\sigma_2^2})) \\ &= \int p(x) * (-\log(\sigma_2) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{(x^2 + \mu_2^2 - 2x\mu_2)}{2\sigma_2^2})) \\ &= -\log(\sigma_2) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{(\mu_1^2 + \sigma_1^2 + \mu_2^2 - 2\mu_1\mu_2)}{2\sigma_2^2}) \\ &= -\log(\sigma_2) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{((\mu_1 - \mu_2)^2 + \sigma_1^2)}{2\sigma_2^2}) \end{aligned}$$

Subtracting second term from the first

$$\begin{aligned} KL(P||Q) &= ((-\log(\sigma_1) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{1}{2}))) - (-\log(\sigma_2) + \log(\frac{1}{\sqrt{2\pi}}) - \frac{((\mu_1 - \mu_2)^2 + \sigma_1^2)}{2\sigma_2^2})) \\ &= -\log(\sigma_1) - \frac{1}{2} + \log(\sigma_2) + \frac{((\mu_1 - \mu_2)^2 + \sigma_1^2)}{2\sigma_2^2} \end{aligned}$$

Verification of this derivation using sympy can be found here [4]

7. I get a different answer for $D_{KL}(P, Q)$ and $D_{KL}(Q, P)$. How is it possible?

It is supposed to be this way, i.e. KL divergence is asymmetric in nature. Let's try to understand this with the formula. We saw that

$$D_{KL}(P \parallel Q) = H(P, Q) - H(P)$$

By that definition,

$$D_{KL}(Q \parallel P) = H(Q, P) - H(Q)$$

We can see that the second term, $H(P)$ and $H(Q)$ should give us completely different values

8. Could you share a code snippet for calculating KLD in python (numpy)?

Here you go! [5]

```
def kld(y_true, y_pred):  
    """Computes Kullback-Leibler divergence between `y_true` and  
    `y_pred`.  
    `loss = y_true * log(y_true / y_pred)`  
    """  
    return np.sum(y_true * np.log(y_true / y_pred))
```

Project Implementation

KL Divergence is the one of the most popular and effective metrics to quantify the similarity of two probability distributions. It has its use cases in a variety of fields such as computer vision, information systems, document retrieval and speech recognition to name a few [8]. It has also been used as a learning mechanism, for example, as a loss function to variational autoencoders (VAE). In this section, we will explore this specific application, wherein we will train a simple VAE with the help of KL divergence for an image generation task.

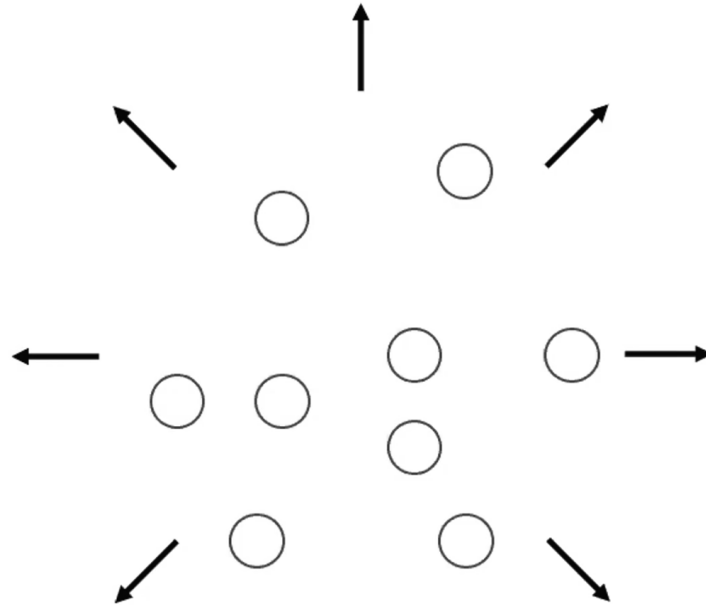
Let's formally define our problem statement. We train a deep learning model, more specifically, a multilayer perceptron based variational autoencoder to generate a 28x28 image resembling that of Fashion MNIST [9]. The brief explanation of the inner workings of VAE [11] is as follows -

- VAE is made up of two parts, an encoder part and a decoder part
- The encoder part is responsible to approximate the mean and the variance of the training data
- Whereas the decoder part tries to reconstruct the image back from the output of our encoder
- To introduce variability in generation, we take a random sample from the mean and variance of the encoder before passing it to the decoder
- The loss function of VAE is comprised of reconstruction term (based on cross entropy loss) and regularization term (based on KL Divergence)

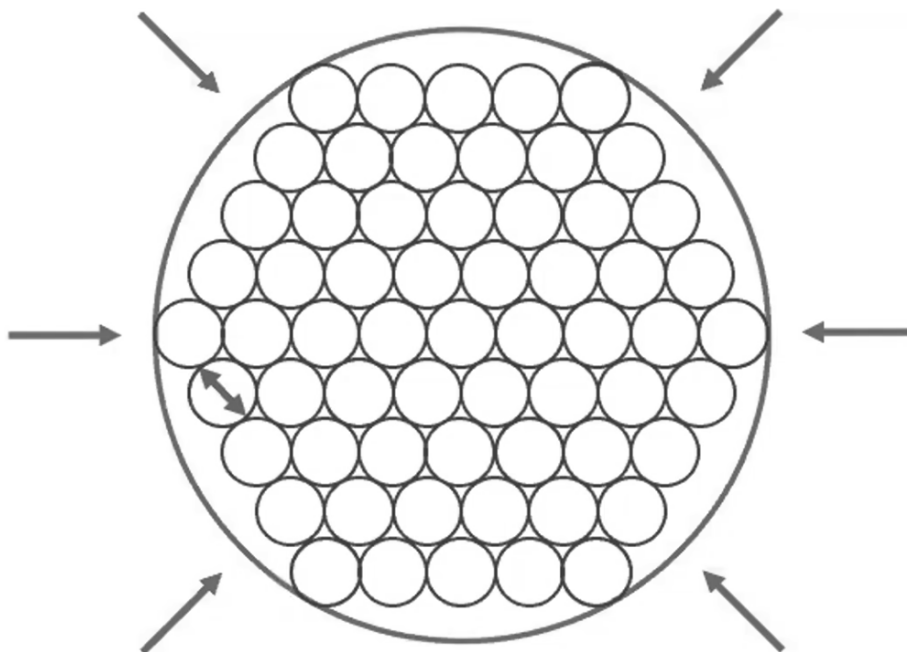
Let us expand on the use of KL divergence in VAE -

The default behaviour of the regularization term (i.e. without using KLD) is that it tends to define each image as a distinct entity, and in that pursuit, it fails to understand the relationship between the images in higher dimension.

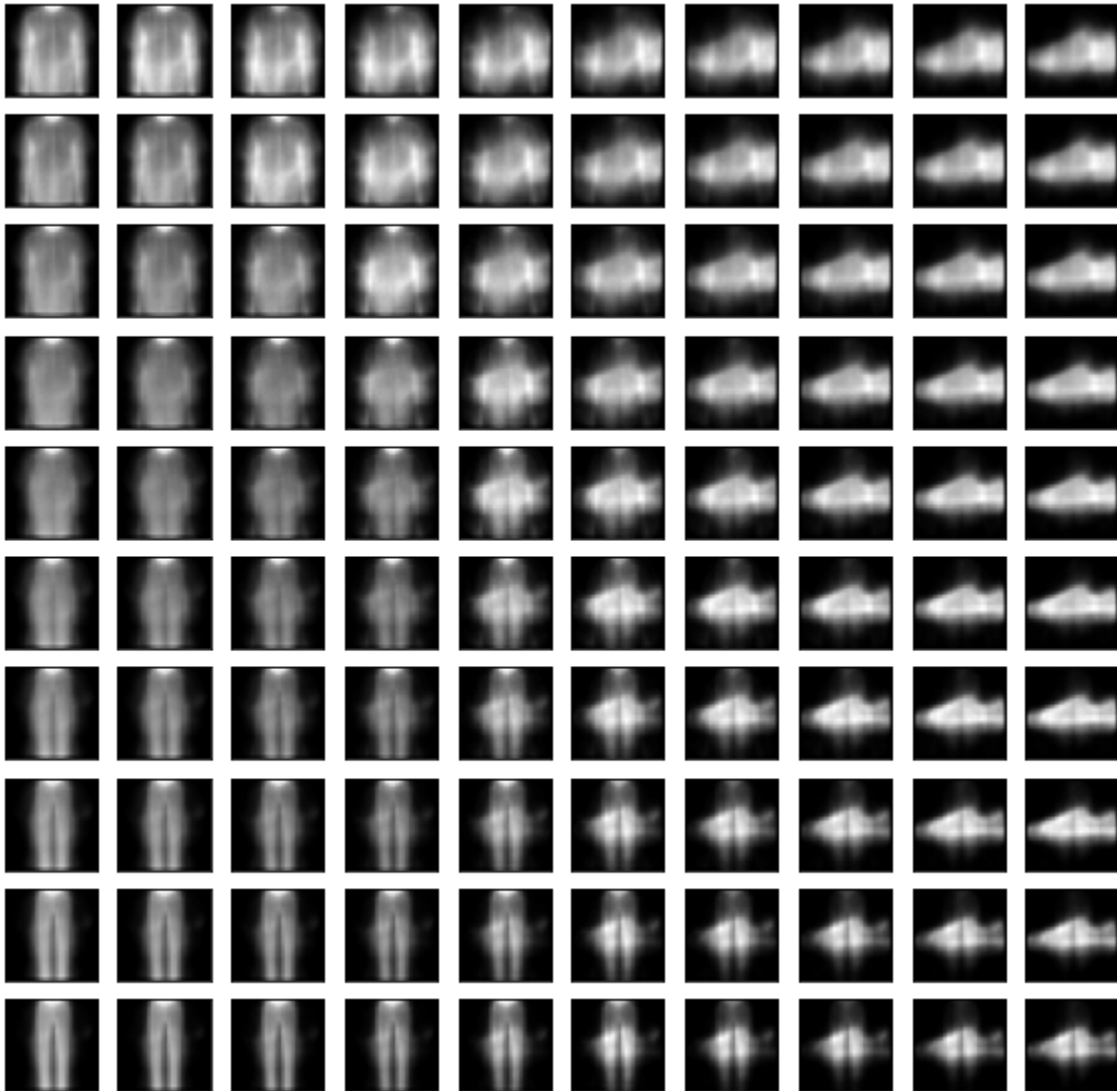
Suppose the image below represents estimated values as circles, wherein the centre is the mean, and the radius is the variance [7]. The regularization term pushes the estimated region away from each other so much that the training collapses.



In order to penalize this, we introduce a regularization term which keeps the bounds intact. This is done by minimizing the KL Divergence between the estimated region and a univariate gaussian with mean 0 and variance as 1. This prevents the explosion of the loss function of VAE.



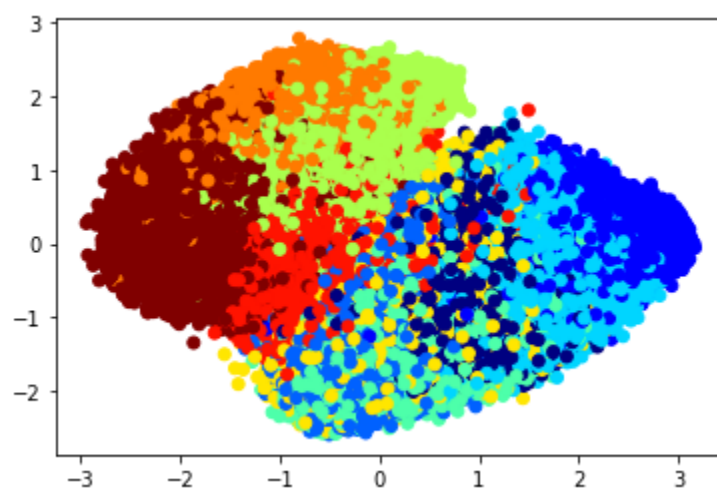
Results




```

Epoch 1/10
469/469 [=====] - 4s 5ms/step - loss: 83.8903 - reconstruction_loss: 82.6529 - kl_loss: 1.2375
Epoch 2/10
469/469 [=====] - 2s 5ms/step - loss: 64.3586 - reconstruction_loss: 59.9482 - kl_loss: 4.4105
Epoch 3/10
469/469 [=====] - 2s 5ms/step - loss: 57.8553 - reconstruction_loss: 54.0982 - kl_loss: 3.7571
Epoch 4/10
469/469 [=====] - 2s 5ms/step - loss: 53.8306 - reconstruction_loss: 50.2583 - kl_loss: 3.5724
Epoch 5/10
469/469 [=====] - 2s 5ms/step - loss: 52.2906 - reconstruction_loss: 48.9591 - kl_loss: 3.3315
Epoch 6/10
469/469 [=====] - 2s 5ms/step - loss: 51.5129 - reconstruction_loss: 48.3078 - kl_loss: 3.2051
Epoch 7/10
469/469 [=====] - 2s 5ms/step - loss: 51.0612 - reconstruction_loss: 47.9027 - kl_loss: 3.1586
Epoch 8/10
469/469 [=====] - 2s 5ms/step - loss: 50.8299 - reconstruction_loss: 47.6783 - kl_loss: 3.1516
Epoch 9/10
469/469 [=====] - 2s 5ms/step - loss: 50.4997 - reconstruction_loss: 47.3460 - kl_loss: 3.1537
Epoch 10/10
469/469 [=====] - 2s 5ms/step - loss: 50.3185 - reconstruction_loss: 47.1502 - kl_loss: 3.1683
<keras.callbacks.History at 0x7f65c006b290>

```




```

Original image size (in MBs) : 358.88671875
Compressed data size (in MBs) : 0.457763671875

```

Conclusion

References

- [1] Bishop C. (2006). *Pattern Recognition and Machine Learning*
- [2] https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence
- [3] <github link for example code>
- [4] <github link for KLD deriv in sympy>
- [5] <github link for KLD in numpy>
- [6] <github link for VAE>
- [7]  Week 8 – Practicum: Variational autoencoders
- [8] Ji, Shuyi et al. “Kullback-Leibler Divergence Metric Learning.” *IEEE transactions on cybernetics*, vol. PP 10.1109/TCYB.2020.3008248. 28 Jul. 2020, doi:10.1109/TCYB.2020.3008248
- [9] Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Han Xiao, Kashif Rasul, Roland Vollgraf. arXiv:1708.07747
- [10] Kullback, S.; Leibler, R.A. (1951). "On information and sufficiency". *Annals of Mathematical Statistics*. 22 (1): 79–86. doi:10.1214/aoms/1177729694. JSTOR 2236703. MR 0039968.
- [11] Kingma, Diederik P.; Welling, Max (2014-05-01). "Auto-Encoding Variational Bayes". arXiv:1312.6114