Repro-VAE Scene Detection in Anime

Dijing Zhang
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213
dijingz@andrew.cmu.edu

Siqiao Fu
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213
siqiaof@andrew.cmu.edu

Yiping Dong
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213
yipingd@andrew.cmu.edu

Zhengyang Zou
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213
zhengyaz@andrew.cmu.edu

Abstract

2D animation has become a popular category in the field of films. With the help of digital tech, artists are able to comprise great masterpieces in fairly short amount of time. Beyond the fine scenery, the meaning inside every frame is vastly explored for various reasons. For example, scene classification is one of the most explored application when it comes to animation, which can facilitate a wide-range of tasks like anime to text, cross anime scene retrieval and human-centric storyline construction. There have been some supervised learning methods that tackle the problem of scene change detection in animation. However, they all require painful pre-processing to get the ground truth of scene change labels, which can only be acquired by long and tedious manual work. Our goal in this article is to come up with an unsupervised model and get the job done. The significant contribution will lie in the elimination of labeling the whole anime as pre-processing. We use VAE as the baseline model. Although VAE may not seem anything related to the topic of scene change detection, it helps compress the image into a lower but more informative latent space as the representation of that certain image. This compressed representation significantly helps in the task of scene change detection, which will be illustrated in the following acticle. Beyond that, we explored a novel training method to regularize the latent space and we call it "reprojection error". During experiments we found that it did improve the accuracy in most cases. The main contribution:1) achieved reasonable scene change accuracy without the help of labeled training dataset; 2) explored VAE with "reprojection" regularization term, the repro-VAE we create seems better in the field of representative learning.

1 Introduction

Our project aims at the problem of scene detection in 2D anime using unsupervised learning. Specifically speaking, we want to detect whether there is a scene change between consecutive frames in the animation.

Considerable efforts have been put into solving scene detection problems in the field of supervised learning. The problem can be treated as a binary classification model, with inputs as two images and outputs 0 or 1 as identification of scene change between them. However, training a supervised model requires manual annotation of the video first, and manual annotating the video requires a lot of time and energy, which makes unsupervised learning a better solution to this kind of problem.

However, the possibility of achieving a similar effect using unsupervised learning methods has not been fully explored. The most straight-forward solution most people could come up with is to directly measure the L1/L2 distance between consecutive frames, and pick the top N pairs as the most likely N scene changes, because, apparently, scene changes usually come with substantial changes of pixel channel values in the image. But it's not exactly the case. As illustrated in figure 1, there is a scene change between frames 14 and 15. But according to the result of L1/L2 distance, the most likely scene change occurs between frames 10 and 11. This example shows that sometimes it is not sufficient to rely only on the straight-forward L1/L2 measurement to detect scene changes. We need more informative features rather than raw RGB data.

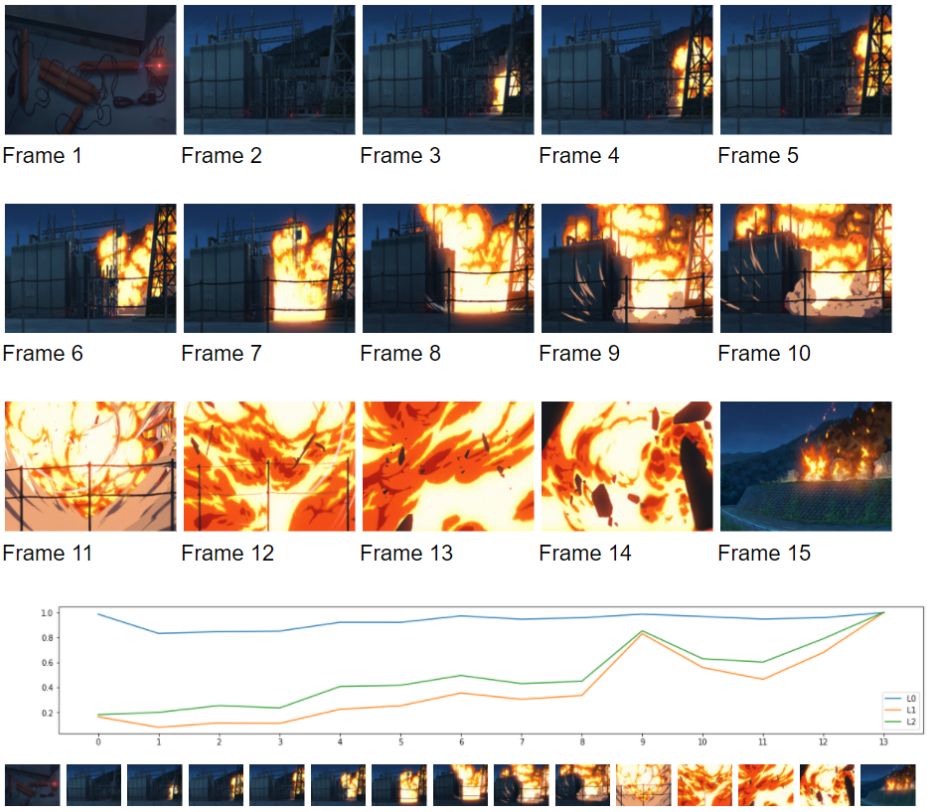

Figure 1: Scene with Fast Changes

With that being said, we try to model this problem using a β-VAE model to learn the latent space distribution of frames in the animation. First, we train our β-VAE model to reconstruct frames. Then the latent vector will be used to generate similarity information between consecutive frames. The latent vectors are expected to be more compact and informative than the raw RGB data, thus generating better scene change detection results.

The key innovation of our methodology is that we do not train the network to perform scene change detection. We only train it to reconstruct images while learning the latent space distribution. Then we apply scene detection as an evaluation metric of our learned latent space.

2 Literature Review

Scene detection and scene classification have been key fields of machine learning research. The majority has been done with the supervised learning method. For example, [1] employs ResNet for video image training and learning to classify video frames into different categories with ground truth labels. [2] proposed a local-to-global scene segmentation framework to cover hierarchical temporal

and semantic information. Both trained a neural network capable of representing the semantics of scene images via supervised learning.

Researchers also investigated the unsupervised learning method to get the work done. [3] proposed a solid way to get general latent space to represent semantic information of input data. With the same methodology, we can perform tasks like scene detection with general latent representation. Furthermore, we can even extrapolate to the task of interpolation of video frames like what is done by [4].

A feasible method to achieve this goal is Variation Autoencoder [5]. Beyond that, researchers are also trying to explore a relationship between beta and latent variables it creates [6]. However, there are also some problems needed to be solved such as its low reconstruction quality. So a neural architecture search might be helpful on that [7].

Thanks to the previous final report and Konan's reference, these are quite helpful to our review and proposal. [8], [9]

3 Contribution

3.1 Reasoning behind the model

As stated repeatedly above, we utilize Variational Auto Encoder as our unsupervised model, but not yet bother to explain the reason behind. What VAE can do is to compress the image into a lower latent space representation through multiple convolutional layers, which is called encode process. Then project the compressed latent vector back to the data space through multiple layers of transposed convolutional neural network, which is called decode process.

VAE is a popular option in the task of generating model. The inference only involves the decoder part, where you generate a latent vector manually and decode a data instance. But in our task, it is quite a different story. We only take the part of encoder into use during inference. More specifically, we put each validation image through our trained encoder, and get a latent distribution for each data instance. We use that distribution as the latent representative of each image, which gives us a way to measure the similarity of two images in a more informative way.

3.2 Mathematical Details

3.2.1 Loss Function of a VAE

The information content of a probability distribution can be quantified as $-\log P(x)$. Entropy refers to the expectation of this quantifiable information with respect to the same probability distribution, given as $-\log P(x)$.

Relative Entropy or the Kullback-Leibler Divergence (henceforth referred to as KL divergence) is a measure of dissimilarity between two distributions. Mathematically, it can be expressed as:

$$KL(P||Q) = -\sum_{x \in X} P(x) \log Q(x) + \sum_{x \in X} P(x) \log P(x) \tag{1}$$

In simpler terms we can write,

$$KL(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)} \tag{2}$$

Note that KL Divergence has two characteristics:

- It is always non-negative, i.e. $KL(P||Q) \geq 0$
- It is asymmetric, i.e. $KL(P||Q) \neq KL(Q||P)$

It is because of its asymmetric nature it is referred to as "divergence" and not distance.

Given a distribution $P(x)$, we aim to know the distribution of z, i.e., $P(z|x)$. Using Bayes Theorem, we can write,

$$P(z|x) = \frac{P(x|z)P(z)}{P(x)} \tag{3}$$

$P(x)$ given by $\sum_z P(x|z)P(z)$ is intractable which in turn makes computation of $P(z|x)$ intractable. One way to compute $P(z|x)$ is using variational inference. We approximate the true $P(z|x)$ by means of a tractable distribution $Q(z)$. We aim to learn a distribution $Q(z)$ which is close to $P(z|x)$, hence minimazing the KL divergence between the two, i.e.,

$$\min_z KL(Q||P) = \sum_z Q(z)\log\frac{Q(z)}{P(z|x)} \tag{4}$$

Using (3) this can be simplified to:

$$\min_z KL(Q||P) = \log P(x) - \sum_z Q(z)\log\frac{P(x,z)}{Q(z)} \tag{5}$$

This can be re-arranged as:

$$\log P(x) = \min_z KL(Q||P) + \sum_z Q(z)\log\frac{P(x,z)}{Q(z)} \tag{6}$$

Let

$$L = \sum_z Q(z)\log\frac{P(x,z)}{Q(z)} \tag{7}$$

There are two points worth nothing in (6) and (7):

- $\log P(x)$ is constant as distribution of x is known. We want to minimize $KL(Q||P)$, hence we should maximize L.
- As $KL(Q||P) \geq 0$, this implies $L \leq \log P(x)$. L is called the Variational Lower Bound.

The problem now reduces to:

$$\max_z L = \max_z \sum_z Q(z)\log\frac{P(x,z)}{Q(z)} \tag{8}$$

Further simplification using (4) yields:

$$L = E_{Q(z)}\log P(x|z) - KL(Q(z)||P(z)) \tag{9}$$

Equation (7) refers to the loss function of the variational auto-encoder where we aim to maximize $E_{Q(z)}\log P(x|z)$ and minimize $KL(Q(z)||P(z))$.

With respect to our implementation,$KL(Q(z)||P(z))$ is approximated by assuming $P(z)$ to be a Gaussian with mean 0 and variance 1. For $Q(z)$, the mean and variance are calculated as encoder outputs. A mathematical analysis yields $KL(Q(z)||P(z))$ as:

$$\mathcal{D}_{KL}(\mathcal{N}((\mu_1,...,\mu_k)^T, \mathrm{diag}(\sigma_1^2,...,\sigma_k^2))||\mathcal{N}(\mathbf{0},\mathbf{I})) = \frac{1}{2}\sum_{i=1}^{k}(\sigma_i^2 + \mu_i^2 - \ln(\sigma_i^2) - 1) \tag{10}$$

$E_{Q(z)}\log P(x|z)$ is the expectation of the $\log P(x|z)$ wrt $Q(z)$. For our model we assume, x to have come from a Bernoulli distribution which gives the Cross-Entropy Loss. It is defined as the expectation of the information in $P(x|z)$ with respect to $Q(z)$. Here, x_n is an input instance and $\hat{x}_n$ is the VAE output.

$$\mathcal{L}_n(x_n,\hat{x}_n) = -[\hat{x}_n\log x_n + (1-\hat{x}_n)\log(1-x_n)] \tag{11}$$

3.3 Algorithmic Details

Based on the basic beta-VAE algorithm and its architecture, we design our own VAE, we call it "Repro-VAE'. Its main contribution is input the reconstructed image to the previous encoder, just like a reprojection procedure. What we want is that the distribution of latent space generated by original image approaches the one of latent space generated by reconstructed image. It kinds like the regularization term of beta-VAE, hope the distribution of z given x be similar to the assumed distribution of z, which is a normal distribution. By adding the reprojection term, these two latent space can be converged and it will contain more representative information. Here is a visualization diagram, explaining what we do.

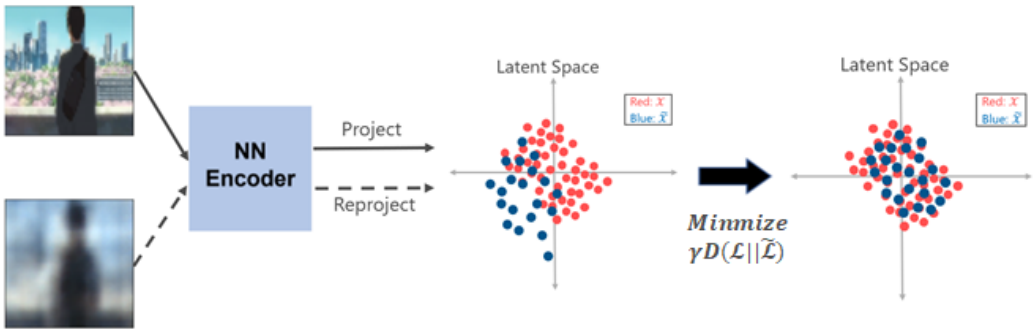

Figure 2: Reprojection Procedure

3.4 Code Implementation Details

Our code implementation is based on this open source GitHub Beta-VAE repository [10]. To better reproduce the results given in our report, several revisions should be made based on the original Beta-VAE model. For hidden dimensions in encoder, we use 4 layers in total ([32,64,128,256]) which is 1 layer less than the original model. In each module in the encoder, we add a mean pooling layer with kernel size of 2 after the 2D convolution layer. Due to that revision, the input size of the mean and variance linear layer should be modified correspondingly. In the decoder, we delete the BatchNorm2d layer and switch LeakyRelu to Relu. In the final layer, we remove the BatchNorm2d layer and Tanh function. As the change the depth of encoder, the resize step in decode function should also be changed. As a new part named re-projection loss is add in our loss calculation, we should make 2 revisions to apply this change. In forward function, after we get the decoded result, mean and variance, we will encode the decoded result and get another pair of mean and variance. In loss function, we receive these 2 pairs of mean and variance and calculate the KL divergence between these two distributions. A self-defined coefficient is needed on this loss when calculating the total loss. This coefficient can be a hyper-parameter similar to beta that can be tuned via experiment.

4 Experimental Evaluation

4.1 Experiments

Table 1: Minor experiment information

model training hardware	Nvidia Tesla V100 SXM2 16GB / Tesla P100 PCIe 16 GB
model inference hardware	Nvidia Tesla V100 SXM2 16GB / Tesla P100 PCIe 16 GB
batch size	256 * 0.05 MB = 12.8 MB
number of worker processes	2
collation function	default
sampling method	random sampling

VAE can be used as the unsupervised learning solution here, because the encoder part can be thought as a type of embedding or feature extractor with input of scene frames. It captures and compresses the features of raw image, to generate a more informative while lower dimensional latent representation. Thus we have a decent embedding to measure the similarity of scenes in two frames.

The main challenge we faced during our experiments is that β-VAE has multiple hyper-parameters to tune for the best scene detection performance. We listed some of the most significant factors as followed, latent space dimension, β, pooling layer type and activation function choice.

We listed different value or choice possible for each hyper-paramters to try with. For example, latent dimension: 5, 10, 20, 50, 100, 500, 1000; β: 0, 0.5, 1, 2, 4; pooling layer: no pooling, mean pooling, max pooling; activation function: ReLU, LeakyReLU, ELU. Obviously, the time required to try every combination of these hyper-parameters is way beyond our project expectation. Instead of trying

every combination, we decided to experiment with one single hyper-parameter a time. To do that, we defined our base-model with the combination of: latent dimension = 10, β = 1, without pooling layer, and use LeakyReLU as activation. Each time when we experiment with one hyper-parameter, we set the other the same as this base-model. Thus we can get the tendency of tuning that particular hyper-parameter to some extent and try to analyze the reason behind if possible.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 64, 64]             896
         AvgPool2d-2          [-1, 32, 32, 32]               0
         LeakyReLU-3          [-1, 32, 32, 32]               0
            Conv2d-4          [-1, 64, 32, 32]          18,496
         AvgPool2d-5          [-1, 64, 16, 16]               0
         LeakyReLU-6          [-1, 64, 16, 16]               0
            Conv2d-7         [-1, 128, 16, 16]          73,856
         AvgPool2d-8          [-1, 128, 8, 8]               0
         LeakyReLU-9          [-1, 128, 8, 8]               0
           Conv2d-10          [-1, 256, 8, 8]         295,168
        AvgPool2d-11          [-1, 256, 4, 4]               0
        LeakyReLU-12          [-1, 256, 4, 4]               0
           Linear-13                [-1, 10]          40,970
           Linear-14                [-1, 10]          40,970
           Linear-15              [-1, 4096]          45,056
  ConvTranspose2d-16          [-1, 128, 8, 8]         295,040
             ReLU-17          [-1, 128, 8, 8]               0
  ConvTranspose2d-18          [-1, 64, 16, 16]          73,792
             ReLU-19          [-1, 64, 16, 16]               0
  ConvTranspose2d-20          [-1, 32, 32, 32]          18,464
             ReLU-21          [-1, 32, 32, 32]               0
  ConvTranspose2d-22          [-1, 32, 64, 64]           9,248
             ReLU-23          [-1, 32, 64, 64]               0
           Conv2d-24           [-1, 3, 64, 64]             867
           Conv2d-25          [-1, 32, 64, 64]             896
        AvgPool2d-26          [-1, 32, 32, 32]               0
        LeakyReLU-27          [-1, 32, 32, 32]               0
           Conv2d-28          [-1, 64, 32, 32]          18,496
        AvgPool2d-29          [-1, 64, 16, 16]               0
        LeakyReLU-30          [-1, 64, 16, 16]               0
           Conv2d-31         [-1, 128, 16, 16]          73,856
        AvgPool2d-32          [-1, 128, 8, 8]               0
        LeakyReLU-33          [-1, 128, 8, 8]               0
           Conv2d-34          [-1, 256, 8, 8]         295,168
        AvgPool2d-35          [-1, 256, 4, 4]               0
        LeakyReLU-36          [-1, 256, 4, 4]               0
           Linear-37                [-1, 10]          40,970
           Linear-38                [-1, 10]          40,970
================================================================
Total params: 1,383,179
Trainable params: 1,383,179
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 8.63
Params size (MB): 5.28
Estimated Total Size (MB): 13.95
----------------------------------------------------------------
```

Figure 3: our final model architecture and size

The above figure is our final model architecture and model size output from torchsummary. It is quite a small model in terms of number of parameters, only takes 13mb to store. It's quite exciting to see how we can solve this problem with such a small model.

4.2 Dataset

The movie **Kimi no Na wa** [11] is our dataset,which has around 370K frames and a csv file including unique frames.

The size of the datasets we will choose from is as follows:

We use 142p as our dataset in the project. The source image size is 189×142, which will be re-scaled into 64×64. Normalization or data augmentation methods are avoided for better reconstruction performance, which will be further illustrated in section 5. As this is a typical image-input pipeline, no special care needs to be taken for dataloader implementation. We just use torch ImageFolder API to load the images.

Table 2: Your Name (Kimi no Na wa) Data Source

Resolution	File Type	Total Size	Avg. Img. Size
142p	png	8.7GB	0.07 MB
213p	png	13.8GB	0.09 MB
320p	png	35.6GB	0.22 MB
480p	png	56.1GB	0.35 MB
720p	png	112.0GB	0.70 MB
1080p	png	217.7GB	1.36 MB

4.3 Evaluation Metrics

The evaluation metrics are quite straight-forward. Given that the trained VAE model encoder is capable of projecting data instance to latent space. We can utilize the latent space representation of each image as some kind of embedding, with which we can measure the closeness of a consecutive pair of frames.

As the latent distribution output by the encoder is a normal distribution, it is nature to come up with KL divergence as the criterion to calculate the similarity of two normal distributions. It's exactly what we do during the training process of VAE, where we calculate the KL divergence between our latent distribution generated with the standard multi-variant normal distribution. The minor difference is that we calculate the KL divergence between two normal distributions generated by consecutive frames.

4.4 Description of Baseline

4.4.1 Baseline Selection

We choose VAE an extra β parameter as our baseline model, also known as β. CelebA is a large-scale face attribute dataset with 202,599 face images, 5 landmark locations ,and 40 binary attributes annotations per image. It is one of the most popular open-source datasets in the field of generative networks. Some reconstruction results are shown below. As we can see here, beta-VAE can have disentanglement, which means the values of latent space represent different components, like the long hair, the color of skin.

Figure 4: Reconstruction qualitative results of Beta-VAE

The baseline beta-VAE has 4 layers of CNN in the encoder network. Feature channels are 32, 64, 128, 256, each with kernel size 4, stride 2, followed by BatchNorm. The activation function is LeakyReLU. The latent space dim is set to be 100. As for the decoder part, it uses symmetrical architecture as the encoder part. The only difference is that there is no BatchNorm or activation function after the final Conv layer in the decoder.

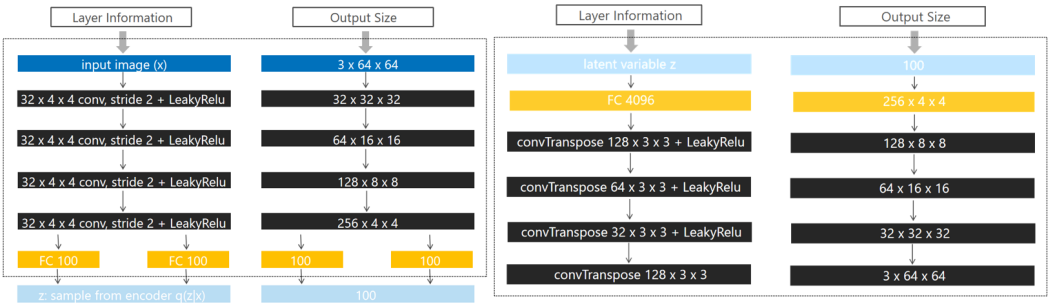

Figure 5: Baseline model architecture

It has been proved in [12] that the baseline model performs reasonably well on CelebA dataset. Applications like facial attribute prediction, facial attribute manipulation can be done by the learned latent space distribution. So we decide to select the exact same architecture as our baseline model and see how it performs on our dataset.

4.4.2 Normalization

Batch normalization and normalization are common-used techniques in CNN application and they often lead to better results in many fields, like classification and so on. However, in our case, batch normal and normal tend to cause great decrease in accuracy, so they are not adopted in our proposed model architecture. Batch normalization is first introduced in [13] and designed to reduce internal co-variate shift. It turns out that batch normalization can reduce the gradient vanishing problems thus makes training much faster. But to our image to image task, the absolute difference of data that BN reduces is much more important than in other tasks. BN makes the network invariant to data re-centering and re-scaling, our task is a regressing task and the target outputs, which are the reconstruction images, are highly correlated to inputs first order statistics.[14]

4.4.3 Beta-VAE Architecture

The model at the core of this work is a Beta-VAE architecture. This is a one-to-one mapping, which means for each input image we are generating one latent space and one reconstructed image. Beta-VAE makes the latent space learned by the encoder to be uncorrelated. The learning process in beta-VAE is representation disentanglement and through this, we can use unsupervised learning to learn the latent features of individual frames, independently.

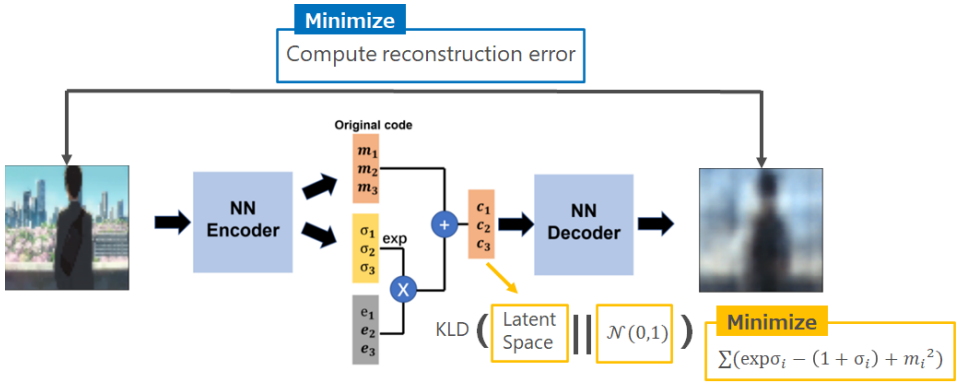

Figure 6: Beta-VAE Diagram

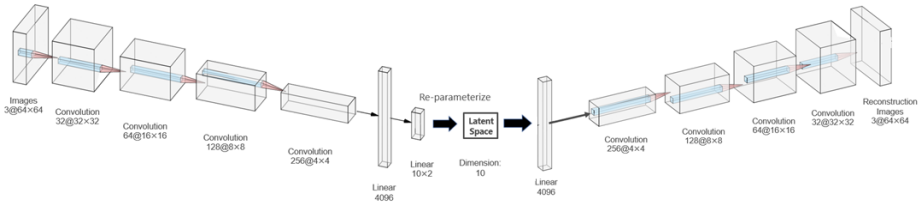

Figure 7: Beta-VAE Architecture

4.4.4 Repro-VAE Loss Function

We want to develop an unsupervised deep generative model that, using samples from X only, can learn the joint distribution of the data x and a set of generative latent factors z such that z can generate the observed data x. Thus, a suitable objective is to maximize the marginal (log-) likelihood of the observed data x in expectation over the whole distribution of latent factors z:

$$\max_\theta E_{p_\theta(z)}[p_\theta(x|z)]$$

For a given observation x, we describe the inferred posterior configurations of the latent factors z by a probability distribution $q_\theta(z|x)$. If we set the prior to be an isotropic unit Gaussian $p(z) = \mathcal{N}(0, 1)$, hence arriving at the constrained optimization problem, where specifies the strength of the applied constraint.

$$\max_{\phi,\theta} E_{x\sim D}[E_{q_\phi(z|x)}[\log p_\theta(x|z)]]$$
$$\text{s.t.D}(\text{q}_\phi(\text{z}|\text{x})||\text{p}(\text{z})) < \epsilon$$

Then we can derive the beta-VAE constrained optimization formulation:

$$\mathcal{L}(\theta, \phi; x, z, \beta) = E_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta D(q_\phi(z|x)||p(z))$$

The first term on the right accounts for the reconstruction error, and the second term represents the similarity of our latent space distribution we get from our model with the "True" latent space distribution, which is assumed to be a multivariate normal distribution. We will try different reconstruction loss functions, such as Binary Cross-Entropy and Mean Squared Loss, while sticking to Kullback-Leibler divergence between two distributions for the second loss term.

After we adding reprojection technique, we need to add one more term to basic loss function, which we call "reprojection error":

$$\mathcal{L}(\theta, \phi; x, z, \beta) = E_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta D(q_\phi(z|x)||p(z)) + \lambda D(q_\phi(z|x)||q_\phi(z|\tilde{x}))$$

The reprojection error can be considered as a regularization term. Different from the one of beta-vae, reprojection error aims at applying constraint on the latent space, hoping the latent space generated by original image and the one generated by reconstructed image owns similar distribution, so that the latent space are much more representative.

5 Results

The results of our experiments can be divided into two parts. The first part is called hyper-parameter searching, which is to try to analyze one hyper-parameter at a time. Each model is trained for 20 epochs, for accuracy seems to stay stable after that. The second part is called final model tuning, where we try to utilize what we have found in the first part of experiments and come up with the combination with the highest scene change detection accuracy.

5.1 β Selection

First, we tried different β with other hyper-parameters set still. Although some turbulence did occur around β = 1,2, it is safe to say that the reconstruction performance decreases with larger β, while KLD loss decreases as well, which is expected to be the case. Since the role of β is a weight between the performance of reconstruction and the regularization of generated latent distribution to be like

Table 3: scene detection accuracy of base-model with different β

β	accuracy	recons loss	KLD loss
0	0.081	199.7	329.9
0.5	0.793	203.6	26.8
1	0.803	202.5	22.0
2	0.796	192.5	18.7
4	0.808	225.3	13.3
8	**0.820**	293.3	5.9
16	0.817	360.4	4.3

normal distribution. It obviously can not satisfy both at the same time. However, what is new is that the scene detection accuracy seems to agree with KLD loss to some extent. The highest peak of accuracy occurs at $\beta = 8$, which also means it does not completely agree with KLD loss. But it's good to know that somehow more regularized latent space can generate better scene detection result.

5.2 Latent Space Dimension

Table 4: scene detection accuracy of base-model with different latent dimension

latent dimension	accuracy	recons loss	KLD loss
5	**0.821**	268.4	13
10	0.803	203.6	26.8
20	0.777	166.7	33.7
50	0.776	159.1	43.8
100	0.787	155.1	44.8
500	0.806	165.8	42.5
1000	0.788	165.9	43.8

Then, we experimented with choice of latent space dimension. We increased latent space dimension from 5 all the way up to 1000. From the results of loss, we found that reconstruction loss decreases while latent dimension increases. And KLD loss behaves the opposite. From these results, we can further indicate that there are some relation between the accuracy of scene change detection and the regularization of latent space, which can also be attributed to KLD loss. Because the lower KLD loss is, the higher accuracy it predicts.

5.3 Pooling Layer

Table 5: scene detection accuracy of base-model with different pooling layer

pooling layer	accuracy	recons loss	KLD loss
none	0.803	203.6	26.8
mean pool	0.814	205.1	22.2
max pool	0.813	208.9	21.7

Next, we tried different pooling layers. We tried the most common two options of pooling layers between convolutional neural layers. From what we learned at class, the difference between mean pooling and max pooling is that the former enforces a linear restriction onto the feature map while the latter enforces a non-linear restriction on it. So the question of which pooling option is better is indeed a hard one, because it really depends on the task itself.

In our case, mean pooling and max pooling both improves the model a little in the scene change detection accuracy.

Table 6: scene detection accuracy of base-model with different activation function

tactivation function	accuracy	recons loss	KLD loss
LeakyReLU	0.803	203.6	26.8
ReLU	0.812	201.3	22.3
ELU	0.808	210.4	21.4

5.4 Activation Function

Last hyper-parameter we experimented in the first part of our experiments is activation function. We didn't stick to LeakyReLU, which is the default implementaion of β-VAE. We experimented with ReLU and ELU. The results of ReLU is slightly better than the other two, which, sadly, couldn't be further explained.

5.5 Final Model With Reprojection

After getting all the results from our hyper-parameters searching with controlled variants. We are very close to getting the best combination of hyper-parameters, and see whether it can improve further.

Table 7: scene detection accuracy of model with the separate best hyper-parameters

hyper-parameter	value	accuracy on all scenes	accuracy on top100
β	2	0.78	0.91
latent dimension	5		
pooling layer	mean pool		
activation	ReLU		

However, when we just simply put all the best separate choices together, things get really messed up. We only get an accuracy of 0.77 with β = 8, latent dim = 5, ReLU activation and mean pooling layer in between. The poor result indicates that the assumption of each hyper-parameter only influences the scene change detection performance independently from one other is wrong. They must be somehow coupled with each other, otherwise the combo of separate best choice would yield a accuracy a lot better than 0.82.

With that said, we need to do the hyper-parameter searching all over again, because we don't know how the combination will perform given the controlled variants experiments on each of the hyper-parameter.

Table 8: scene detection accuracy of our final model

hyper-parameter	value	accuracy on all scenes	accuracy on top100
β	4	**0.825**	**0.96**
latent dimension	10		
pooling layer	mean pool		
activation	LeakyReLU		
reprojection KLD loss weight	200		

After long time of experiments, we finally land on the best model we could come up with. This time, we add the reprojection loss term into the picture. The final model parameters are shown in the table above. The scene change detection accuracy of our final model is 0.825, which is slightly higher than everything we tried before taking reprojectiong KLD loss into consideration.

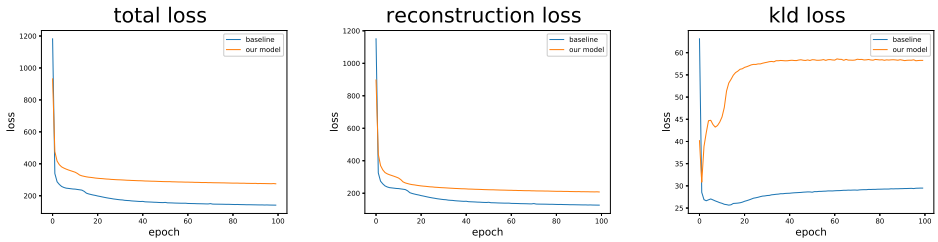

Figure 8: our model versus baseline with $\beta = 0.5$

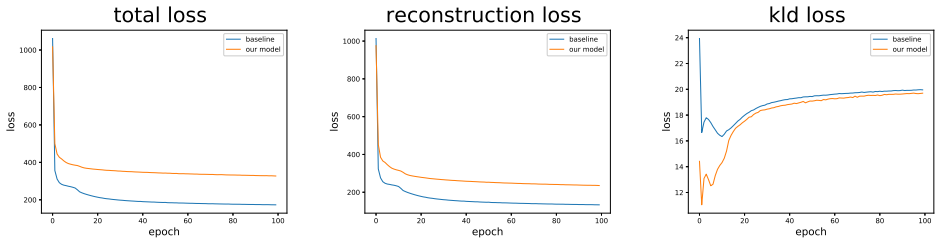

Figure 9: our model versus baseline with $\beta = 2$

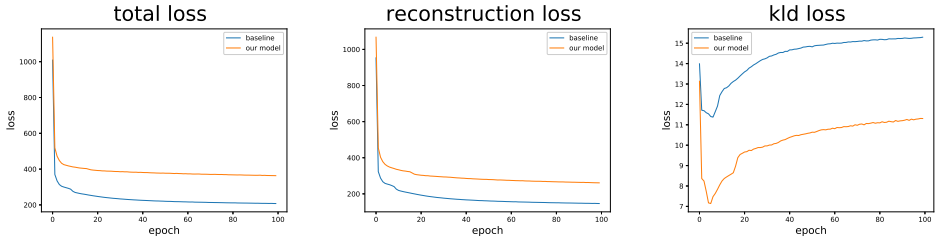

Figure 10: our model versus baseline with $\beta = 4$

Above figures are the total loss, reconstruction loss and KLD loss of our model and baseline model with different β values. Some patents can be concluded from the figures, the reconstruction loss of our model is always higher than baseline model. This is expected since the reprojection KLD loss puts extra restriction on the latent space distribution, which will sacrifice the performance of reconstruction of images. As for KLD loss, when choosing lower β, such as 0.5 or 1, our model has a sudden drop at the early stage. It slowly rises up with further training and converges at a higher value than baseline model does. But for larger β value like 2 and 4, KLD loss of our model is strictly lower than that in baseline model, which further illustrates why we have the best accuracy with $\beta = 4$ in our final model.

5.6 Inference with K-Means

All the inference discussed above is based on the fact that we have some pre knowledge of the total number of actual scene changes in the dataset. All we need from the model output is where these scene changes happens. It is a strong constraint because we usually can not tell the number of scene changes without manually counting and labelling.

With that said, we have to figure out a way to perform scene change detection without the input of number of scene changes. That's where K-means clustering comes into our view. After calculating N-1 KL divergence between each consecutive pairs in N frames, instead of taking the top K values which we did before, we will run K-means on these scalar values with K = 2. Due to the fact that scene changes mean high KL divergence and no scene changes means low KL divergence, we hope that the K-means clustering will automatically draw a line of KL divergence value between scene changes and no scene changes frame pairs.

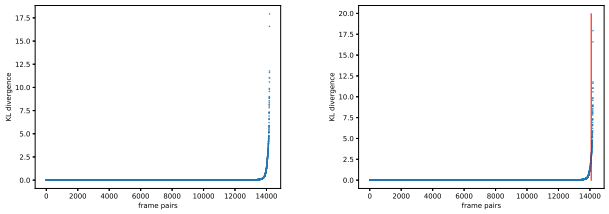

Figure 11: sorted KL divergence on validation set

The figures above is the sorted KL divergence between N-1 consecutive frame pairs. It is very obvious that the majority of KLD value is nearly zero. And around 200 frame pairs can exceed KLD = 1. The red vertical line in the right figure above is the result of K-means when k = 2.

Table 9: F1 score analysis

	Positive	**Negative**
True	110	13934
False	6	147

Given that our validation set is quite imbalanced in terms of positive and negative cases, it is not adequate to analyze the performance of our model with accuracy alone. We calculated the F1 score of out model performance shown in the table above.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1\,score = \frac{2 \times precision \times recall}{precision + recall}$$

According to the formula above, the F1 score of our model is 0.59 when we use K-means to classify scene changes. That's not a high value compared to 0.825 precision. Inference with K-means missed about half of the scene changes, which results in a lower recall rate. The conclusion is that the performance of our model does suffer in the cases that real scene change total number is not available in the way of low recall rate.

One more idea that we can explore in the future. Above we only consider the KLD between latent space and do KMeans in one dimension. However, what about we introduce another metric or more metrics as extra dimensions? So that we can do KMeans in 2D or 3D Dimension. The reason why we want to do this is because we think KLD consider certain information of the scene and some other information will be ignored, so not all scenes can be detected using KLD. We can introduce another metrics as evaluation methods, as a supplement.

5.7 Latent Space Visualization

In our final model, we set latent space dimension as 10, which requires some special metric to visualize the learned latent space representation. We uses t-distributed stochastic neighbor embedding (t-SNE) to compress the latent dimension from 10 to 2, which is a very popular solution in a wide range of applications in signal processing.

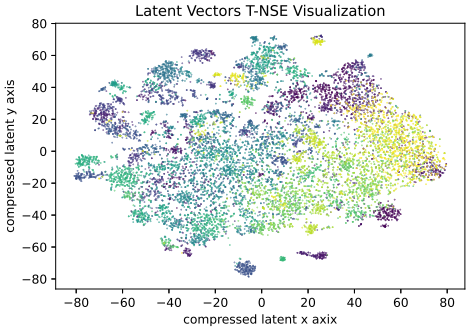

Figure 12: visualization of latent vectors using t-SNE

The above figure shows the result of visualization using t-SNE. We used all 14197 latent vectors from the validation frames, which includes 257 scene changes. We give each batch of scenes without changes the same color. And from the figure, certain pattern can be told that frames without scene changes kind of cluster in the latent space, which further qualifies our idea that the learned latent space can represent information that are adequate to perform scene change detection task.

5.8 Scene Change Detection Visualization

Our accuracy and F1 score seem quite good. But we are still curious about what kind of scene change we cannot detect and what kind of frames we consider there exist change but instead they are not. We can targetedly improve our method. Here are some visualization results.

Figure 13: True Positive Scene Detection

Figure 13 shows the true-positive results of our model, which means we correctly detect the existing scene change. As we can see, it works quite well to these kind of obvious scene change.

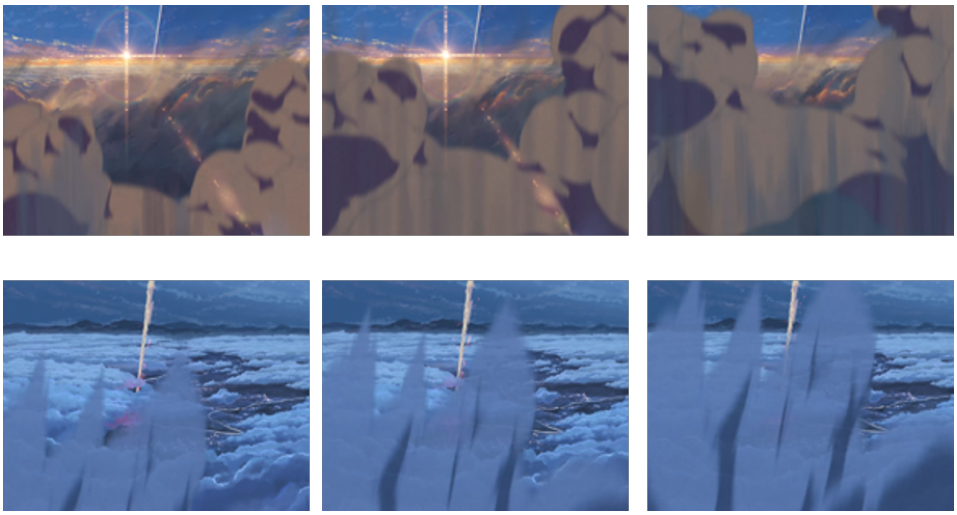

Figure 14: Dynamic Change - False Negative

Figure 14 are quite interesting and it proves the robustness of our model. Like we mentioned before, L1/L2 will fail if there exists dynamic changes in the consecutive frames. However, our model can easily classify these and determine them as no scene change. It is quite an improvement!

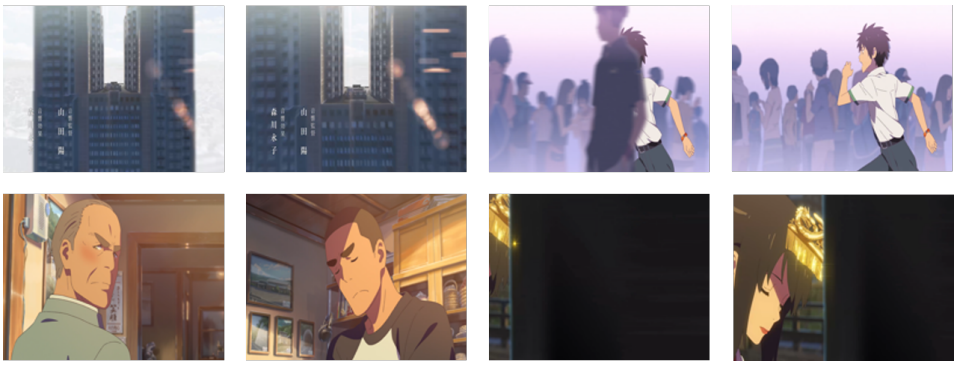

Figure 15: True Negative Scene Detection

Figure 15 shows some true negative scene changes, which means no scene changes exist but model predicts they are. As we can see from the pattern of these images, we can find the most likely reason is the quickly zoom-in or zoom-out of scenes or the appearance and disappearance of main characters. It is quite difficult to deal with the problem, because we didn't apply attention or segmentation to images, what Repro-VAE learned is the implicit distribution of the images, which includes the main characters. We can try self-attention or segmentation parts to our model, try to abandon the problems caused by the zoom-in effect and the presence of characters.

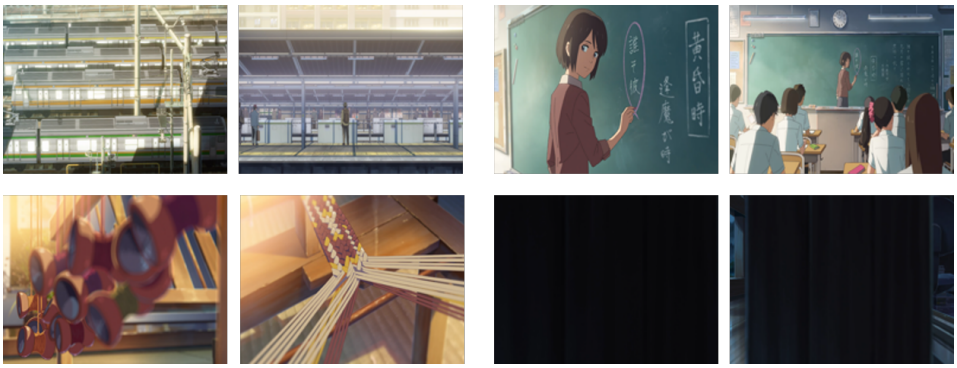

Figure 16: False Positive Scene Detection

Figure 16 presents some false positive scene changes, which means these should be scene changes but the model said they are not. We can find out if the color pattern or whole style are similar, our model cannot predict well. Here also exist zoom-in, zoom-out effect. The solution is not clear right now, but we will try self-attention or introduce other metrics to determine whether there exists a scene change.

The ground truth labels cannot be absolutely right so that doubt may exists whether these are scene change or not. However, it is the motivation we want to design a robust model to correct manual labelling, even take the place.

6 Conclusion and Future Work

The main contribution of our project is to explore how β-VAE can be utilized as an unsupervised learning model to tackle the problem of animation scene change detection. The most exciting part of our project is we create one regularization term, named reprojection loss. So we explore the usage of our model – repro-VAE in this task and achieve a pretty good result.

We first searched for the separate best hyper-parameters, and tried to combine them up to get the best accuracy. However, with further experiments we found that this naive assumption failed. The combination of separately-best hyper-parameters yielded poorer accuracy. This indicates that these hyper-parameters are coupled with one another. Tuning should be done jointly. Given what we found, we jointly tuned hyper-parameters and added reprojection KLD loss term into our training process. The reprojection KLD loss is a weighted KL divergence between the latent distribution in the standard pipeline of β-VAE and the extra latent distribution generated by feeding the reconstructed output back through encoder. The whole process is a reprojection in the latent space, which is why we named it that way. The final model gives out 0.825 accuracy in the scene change detection task, compared to 0.78 with baseline model.

Although we did get higher accuracy out of our final model, the capacity of Repro-VAE is far from fully-explored. The range of our architecture search is limited to certain hyper-parameters without touching on model depth and width due to limited time of our course span. We believed the idea of Repro-VAE can help in various tasks that requires the regularization of latent space, with great potential. And its mathematical theory also needs further exploration. Besides, like self-attention, segmentation model, introducing other metrics as inference can also be a topic we can explore in the furture. We hope to dig into our Repro-VAE and go for a publish.

Finally, thanks Prof. Raj for this excellent course and thanks TA Konan and TA Eason for their kind help in this project!

7 Division of work

All team members reviewed papers and finalized the baseline model and modification plan together. Everyone is in charge of their own coding part in this project and contributed equally to writing the final report and presentation slides. Dijing Zhang additionally did the video presenting and recording work.

References

[1] Jiang, Dayou, and Jongweon Kim. "A scene change detection framework based on deep learning and image matching." *Advanced multimedia and ubiquitous engineering.* Springer, Singapore, 2018. 623-629.

[2] Rao, Anyi, et al. "A local-to-global approach to multi-modal movie scene segmentation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020.

[3] Higgins, Irina, et al. "beta-vae: Learning basic visual concepts with a constrained variational framework." (2016).

[4] Jiang, Huaizu, et al. "Super slomo: High quality estimation of multiple intermediate frames for video interpolation." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018.

[5] Higgins, I. et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." ICLR (2017).

[6] Burgess, C. et al. "Understanding disentangling in β-VAE." ArXiv abs/1804.03599 (2018): n. pag.

[7] Elsken, T. et al. "Neural Architecture Search: A Survey." ArXiv abs/1808.05377 (2019): n. pag.

[8] Mehak Goyal. et al. "Scene Detection in 2D Animation using beta-VAE" S20-final report in 11785

[9] Joseph Konan, "Project Type C: Unsupervised Scene Detection in Anime" Reference draft.

[10] Anand Krishnamoorthy, PyTorch-VAE, (2020), GitHub repository, https://github.com/AntixK/PyTorch-VAE/tree/master/models

[11] Your Name. (, Kimi No Na Wa.). CoMix Wave Films, 2018. https://youtu.be/ZrXwZpXOiVA.

[12] X. Hou, L. Shen, K. Sun and G. Qiu, "Deep Feature Consistent Variational Autoencoder," 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 2017, pp. 1133-1141, doi: 10.1109/WACV.2017.131.

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[14] Fan, Yuchen, et al. "Balanced two-stage residual networks for image super-resolution." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2017.

Appendix

A Reconstructed Images

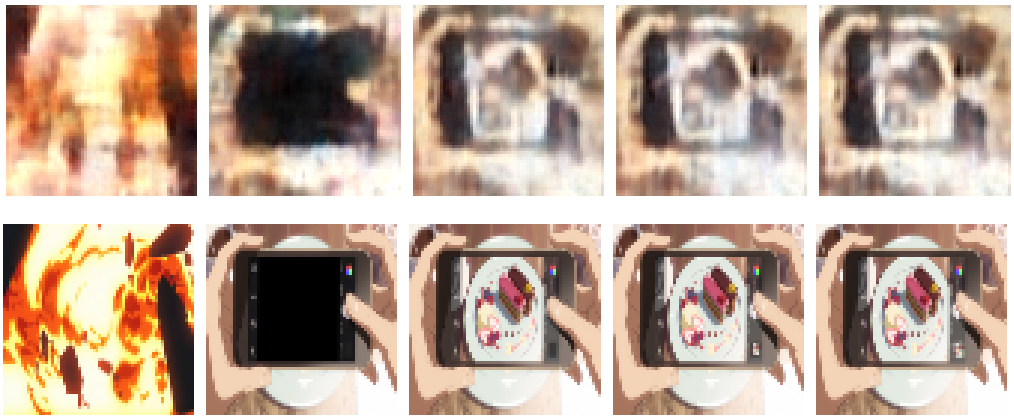

Figure 17: 5 images with highest KLD loss

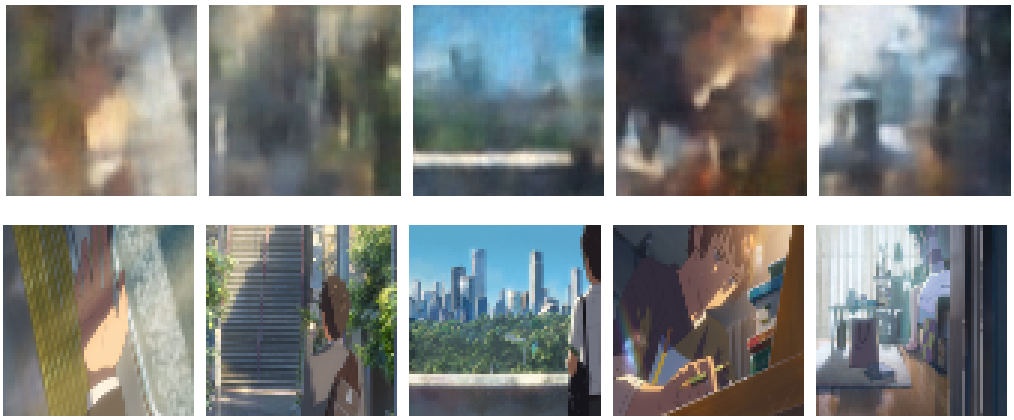

Figure 18: 5 images with average KLD loss

Figure 19: 5 images with lowest KLD loss

Figure 20: 5 images with highest reconstruction loss

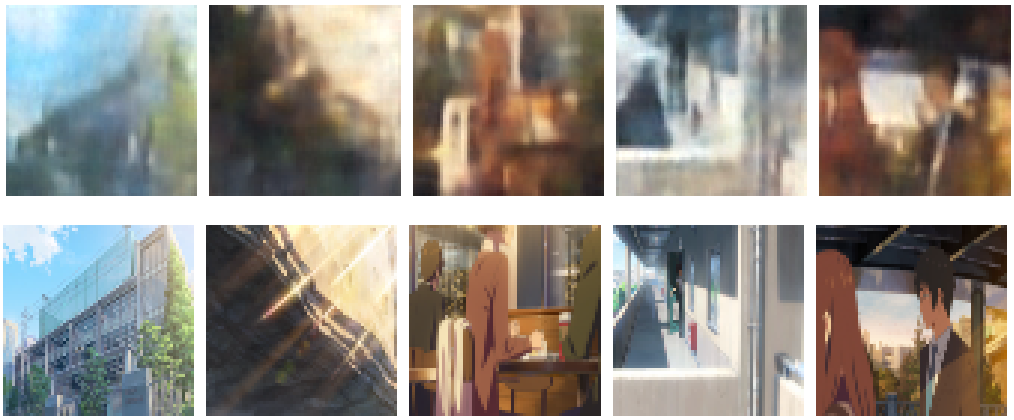

Figure 21: 5 images with average reconstruction loss

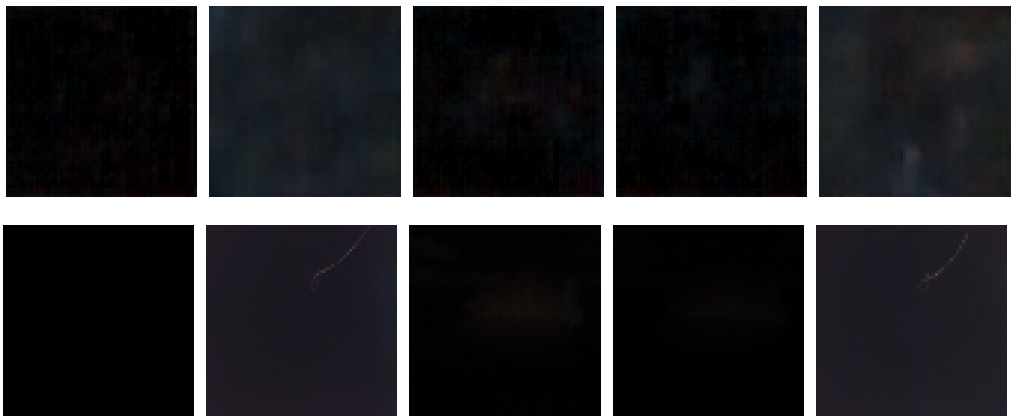

Figure 22: 5 images with lowest reconstruction loss

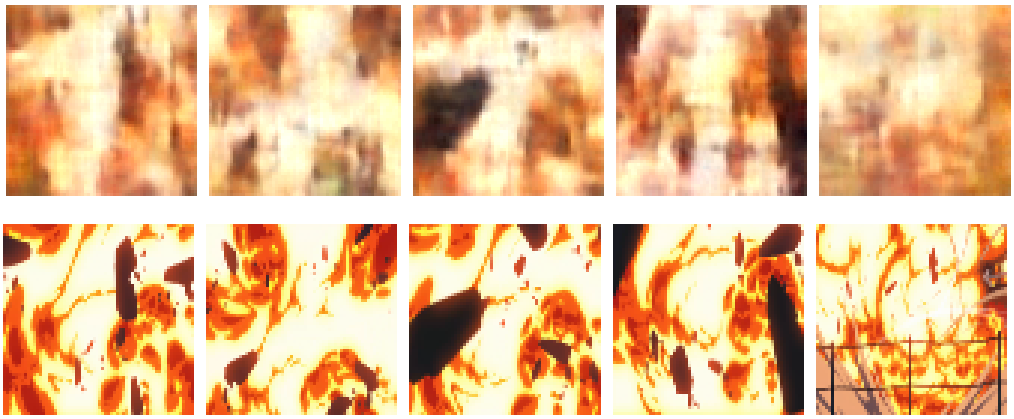

Figure 23: 5 images with highest total loss

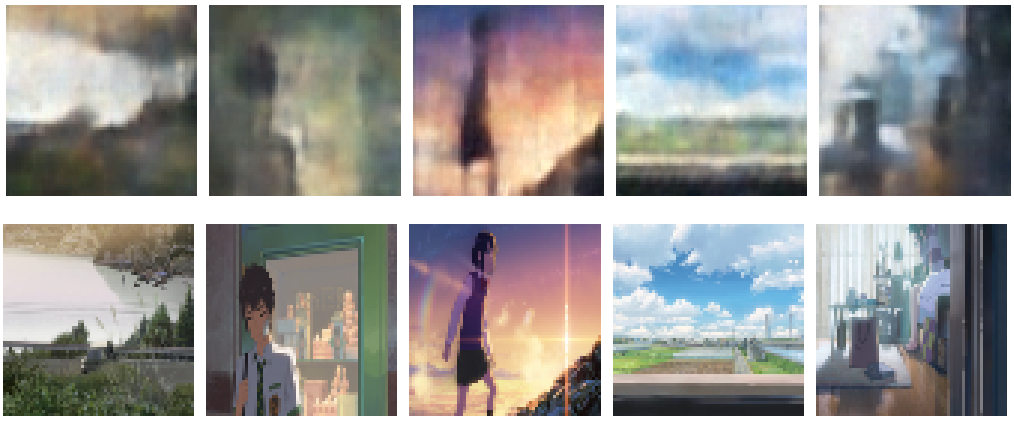

Figure 24: 5 images with average total loss

Figure 25: 5 images with lowest total loss