Autoencoder Project Summary

Yunhao Yang
Department of Computer Science
University of Texas at Austin
Austin, TX 78705
yunhaoyang234@utexas.edu

Yuhan Zheng
Department of Computer Science
University of Texas at Austin
Austin, TX 78705
yuhanzheng@utexas.edu

Abstract

We implement a model that integrates the auto-encoder with latent space clustering. We use a Soft K-means Algorithm to cluster the points in the latent space after the images go through the encoder. For each cluster, there is a filter corresponding to it. After the reconstruction, we use various quality metrics to evaluate the quality of reconstructed images. Currently, the results show that this model generally works better than the single-filter autoencoder.

1 Motivation

Our goal is to restore an image from blurring that is not globally exerted. For instance, video shaking can cause local blurring which is not isotropic in the entire image frame. (i.e. doing image restoration on-demand and locally is non-trivial). We propose to use an attention-based mechanism that manipulates images pixel-wise (or block-wise). Then, the idea of underlying latent space clustering would possibly make a distributional shift under latent space, and when projected back to the image space, the network can generate heterogeneous deblurring results whereas some works treat an image as a unified object. We believe that this approach will be more efficient in runtime since we do not need to apply the deblurring mechanism to every pixel in the image or video. While using various filters rather than one universal filter will result in a more accurate transformation of pixels in the image.

2 Methodology

The model integrates Convolutional Auto-encoder with clustering. It includes an encoder, an decoder, feature space with clustering, and multiple filters (decoders) corresponding to the clusters. The structure is shown in Figure 1.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

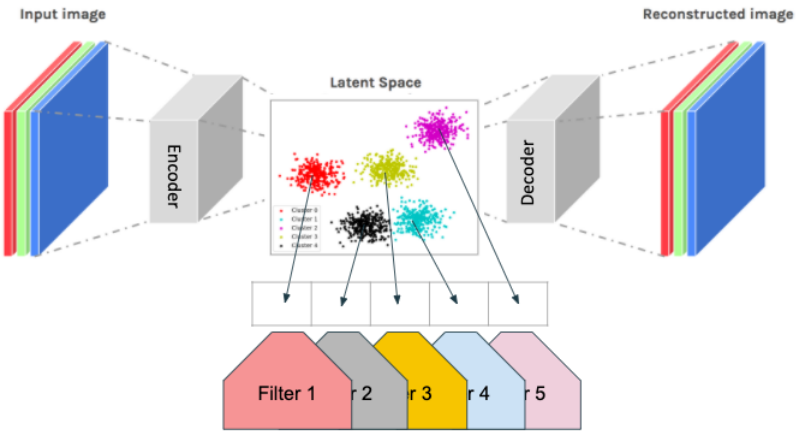


Figure 1: the structure of the multi-filter auto-encoder

2.1 Network Architecture

Our network is implemented based on Convolutional Autoencoder. In addition to the encoder and decoder, there are multiple filters that transform or reconstruct data from the latent space. An images is broken into pixels with neighbors to form blocks and feed them into the network. A pixel and its neighbor pixels (a block) are taken and fed as the input of the encoder. The encoder reduces the dimension of images and sends the dimension-reduced data into the feature space. The encoder consists five convolutional layers, one flatten layer and two dense layers.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 18, 18, 3)]	0
conv2d (Conv2D)	(None, 18, 18, 16)	448
conv2d_1 (Conv2D)	(None, 9, 9, 32)	4640
conv2d_2 (Conv2D)	(None, 9, 9, 48)	13872
conv2d_3 (Conv2D)	(None, 5, 5, 72)	31176
conv2d_4 (Conv2D)	(None, 5, 5, 128)	83072
batch_normalization (BatchNo	(None, 5, 5, 128)	512
flatten (Flatten)	(None, 3200)	0
dense (Dense)	(None, 96)	307296
dense_1 (Dense)	(None, 60)	5820

Figure 2: Encoder Structure

In the feature space, we cluster the dimension-reduced data sent from the encoder using soft K-means algorithm. We use a label vector generated from local features or obtained from public datasets to identify each cluster and determine further actions. Each cluster associated with a filter that transforms pixels, makes a distributional shift under latent space. For example, the filter associated with a "blurred" cluster may shift the pixels from this part of the image into a cluster that is not "blurred".

To achieve it, we construct multiple filters (decoders) each associates with one cluster. Each decoder is trained only using the data from the cluster corresponding to this decoder. Every decoder has the same structure. The decoder consists one dense layer and six convolutional layers. After reconstruction of pixels (blocks), we can merge the blocks together to form the image.

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 60)]	0
dense_2 (Dense)	(None, 1296)	79056
reshape (Reshape)	(None, 3, 3, 144)	0
conv2d_transpose (Conv2DTran	(None, 3, 3, 128)	166016
conv2d_transpose_1 (Conv2DTr	(None, 6, 6, 72)	83016
conv2d_transpose_2 (Conv2DTr	(None, 6, 6, 48)	31152
conv2d_transpose_3 (Conv2DTr	(None, 18, 18, 32)	13856
conv2d_transpose_4 (Conv2DTr	(None, 18, 18, 16)	4624
conv2d_transpose_5 (Conv2DTr	(None, 18, 18, 3)	435

Figure 3: Decoder Structure

Every layer in the encoder and decoder contains a kernel regularizer, which performs regularization on each layer to prevent overfitting. A regularization term (regularizer) $R(f)$ is added to a loss function:

$$min_f \sum_{i=1}^{n} V(f(x_i), y_i) + \lambda R(f)$$

where $V(f(x_i), y_i)$ is the loss function. In this architecture, we use L1-norm and L2-norm to form the regularization term $R(f)$, with specified λ.

2.2 Training Scheme

Before training, we break the training data (images) into fix-sized blocks (pixels with neighbors) and feed the blocks into the autoencoder as individual images.

First, the encoder and decoder are trained to generate better clusters in the latent space. To achieve this, a clustering loss is integrated to the training loss of the autoencoder.

Second, the dimension-reduced data are passed into the z space as vectors that only contain the essential features of pixel neighborhoods according to our trained model. These input data are then classified into clusters based on similarities of features stored in their vectors.

Each cluster is associated with a filter. After the clusters are generated, the filter is trained only using the data that are classified into the corresponding category. This process ensures each filter only does its own transformation without interfered by other data that contain different features.

Each cluster is labeled a vector that contains spatial, spectral, and temporal features of the cluster. In order to deblur the input image, clusters that are identified as blurred will be transformed into corresponding clusters that are identified as higher quality pixels. Clusters are identified respectively to determine whether distribution shift is needed and the exact shift applied. Thus, heterogeneous blurring can be resolved.

2.3 Test and Evaluation

Currently, we use CelebA dataset and divide it into two sets: training set and test set. Each image has the resolution 256×256. Images with high resolutions can be blurred and used as inputs to the auto-encoder. Outputs of the auto-encoder and the original images are used to check for the quality of restoration through Regional PSNR, SSIM, and UQI, which measure the quality of the restored images.

- PSNR (peak signal-to-noise ratio): this ratio is used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image.
- SSIM (structural similarity) is used for measuring the similarity between two images.
- UQI (universal quality index) is a special case of SSIM that all the constants are set to zero.

3 Experiment

3.1 Generating Training Data Set

3.1.1 Divide CelebA into blocks

Images with size 256×256px were selected from the CelebA data set. These images were then split into two data sets: a training set and a validation set. The training set had 225 images, and the validation set had 75 images. Random noise generator were applied to the training set to generate noised images. The noised images have an average PSNR 11.34. Each image, both noised and ground truth, in the training set was then divided into 18×18px blocks, with 4px overlap with the surrounding blocks. For instance, the top 18×4px of the block is overlapped with the bottom 18×4px of the block that sits above the current block. The overlap is applied to four edges of the block in order to reduce the variance between blocks. These blocks were fed into the our network as input data. On merging the blocks, a weight is set for the overlapping area. On the overlapping area, the closer to the block center, the higher weight will be assigned. The blocking artifacts is eliminated.

3.1.2 Data Augmentation

Images with size 256×256px were selected from the CelebA data set. These images were then split into two data sets: a training set and a validation set. The training set had 225 images, and the validation set had 75 images. Data augmentation were applied to the training set: each image was rotated by a random angel. After data augmentation, the training set had 450 images. Bilateral Filtering were applied to the training set to generate blurred images. Each image, both blurred and ground truth, in the training set was then divided into 16×16px blocks.

3.1.3 Resized CelebA images (Used to validate the network)

Images with size 256×256px were selected from the CelebA data set. These images were then split into two data sets: a training set and a validation set. The training set had 11475 images, and the validation set had 825 images. All images were then resized to 32×32px. After that, Bilateral Filtering were applied to the training set to generate blurred images.

3.1.4 Divide MNIST into blocks

MNIST dataset was loaded in as images with size 28×28px. Since all images are black and white, each image had shape (28,28,1). Bilateral Filtering were applied to the training set to generate blurred images. Each image, both blurred and ground truth, in the training set was then divided into 14×14px blocks.

3.2 Building and Training Autoencoder

3.2.1 Basic Network (Convolutional Autoencoder + MSE)

To start the construction of the auto-encoder, an existing autoencoder model was taken to be modified. Pieces of this model were replaced with our architectures: an encoding function, clustering algorithm, and multiple decoding functions.

An encoder was trained with the training set with decreasing learning_rate (start from 0.001 and decrease exponentially 0.9^n), epochs= 100, and batch_size= 128.

Mean Square Error is used to calculate the reconstruction loss. The total loss that used for back-propagation is the sum of the reconstruction loss and kl divergence score.

$$Loss = MSE(x, \hat{x}) + KL(x, \hat{x})$$

where x is the reconstructed image and $\hat{x}$ is the "true label" image. MSE and KL Divergence are defined in Equation 1 and 2. n represents the number of pixels of the image.

$$MSE(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^{n} (x_i, \hat{x}_i)^2 \tag{1}$$

$$KL(x, \hat{x}) = \hat{x} * \log(\frac{\hat{x}}{x}) \tag{2}$$

Then after the latent space was constructed, a clustering function was applied. A soft K-means algorithm was used for this clustering function, and the function was used to generate two clusters from points in the latent space and the labels for each cluster. As we proposed, one cluster should consist of points that represented relatively clear image blocks, and the other one should consist of points that represented relatively blurred image blocks. Each point in the latent space belong to one and only one of the clusters which were associated with distinct labels respectively. Associating labels to clusters in latent space z was done in a semi-supervised manner. After clusters were created, each cluster was used to train one of the filters, so clear and blurred blocks could be treated differently in reconstruction stage. Each filter was trained with the same learning_rate, epochs, and batch_size. The quality of the filter training procedure was evaluated by the chosen loss function, we use MSE (Equation 1) in this case, and the results of image restoration were used to adjust the network.

Along with these two filters, a third decoder was trained with the produced latent space without clustering. This decoder was used for a plain autoencoder that served as the control group for the evaluation stage. The parameters used for training this decoder was the same as the ones used for encoder and the other two filters.

3.2.2 Variational Autoencoder with VGG16

In addition to the network architecture presented in section 2.1, a sample layer is applied to sample the encoder output z based on the mean and variance to construct a Variational Autoencoder. Then, a pre-trained VGG16 network is embedded into the network [10]. VGG16 increases the network complexity in order to be suitable for more diverse image blocks.

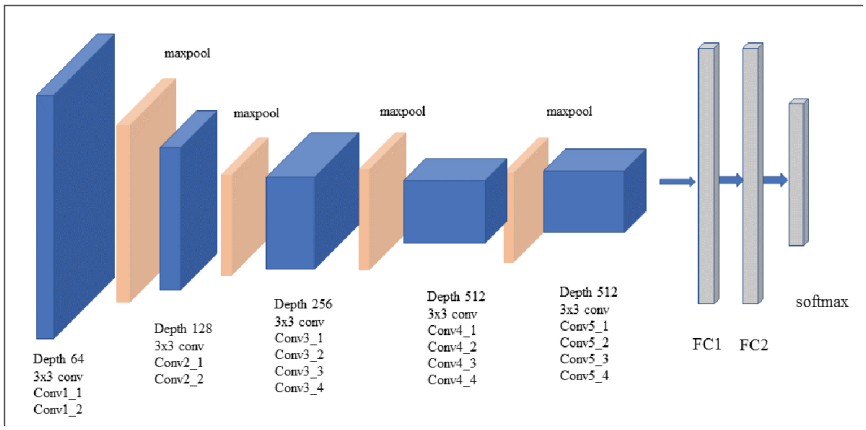


Figure 4: VGG 16 Structure

3.2.3 Variational Autoencoder with ResNet

Residual neural networks (ResNet) utilize skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between [11]. The encoder that presented in section 2.1 was replaced by a simple residual network. The structure of the new encoder is presented in Figure 5.

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	[(None, 16, 16, 3)]	0	
conv2d_4 (Conv2D)	(None, 8, 8, 32)	896	input_3[0][0]
batch_normalization_4 (BatchNor	(None, 8, 8, 32)	128	conv2d_4[0][0]
conv2d_6 (Conv2D)	(None, 8, 8, 32)	896	input_3[0][0]
conv2d_5 (Conv2D)	(None, 8, 8, 32)	9248	batch_normalization_4[0][0]
add (Add)	(None, 8, 8, 32)	0	conv2d_6[0][0] conv2d_5[0][0]
flatten_2 (Flatten)	(None, 2048)	0	add[0][0]
dense_2 (Dense)	(None, 16)	32784	flatten_2[0][0]
z_mean (Dense)	(None, 2)	34	dense_2[0][0]
z_log_var (Dense)	(None, 2)	34	dense_2[0][0]
sampling_1 (Sampling)	(None, 2)	0	z_mean[0][0] z_log_var[0][0]

Figure 5: ResNet Structure

3.2.4 Variational Autoencoder

The Variational Autoencoder consists similar structure with the Convolutional Autoencoder. The sampling layer is added after the last dense layer in the convolutional autoencoder.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 16, 16, 3)]	0	
conv2d (Conv2D)	(None, 8, 8, 32)	896	input_1[0][0]
batch_normalization (BatchNorma	(None, 8, 8, 32)	128	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 4, 4, 64)	18496	batch_normalization[0][0]
batch_normalization_1 (BatchNor	(None, 4, 4, 64)	256	conv2d_1[0][0]
flatten (Flatten)	(None, 1024)	0	batch_normalization_1[0][0]
dense (Dense)	(None, 16)	16400	flatten[0][0]
z_mean (Dense)	(None, 2)	34	dense[0][0]
z_log_var (Dense)	(None, 2)	34	dense[0][0]
sampling (Sampling)	(None, 2)	0	z_mean[0][0] z_log_var[0][0]

Figure 6: VAE Structure

3.2.5 Basic Network with Soft N-Cut Loss

In the clustering step, there is a transformer that transforms the output from the encoder to a cluster-vector. To determine which cluster the block is belong to, simply use $argmax(cluster - vector)$. In order to make clusters separable, a soft n-cut loss[12] is used in the training process. In addition to the Basic Network, a soft n-cut loss is added into the total training loss.

$$Loss = MSE(x, \hat{x}) + \hat{x} \times \log(\frac{\hat{x}}{x}) + soft\_n\_cut\_loss(V, w)$$

In the loss function, x and $\hat{x}$ represent the reconstructed and label images. MSE and KL Loss is defined in Equation 1 and 2. V is the set of all the blocks, p represents the cluster vectors of the blocks and w represents the latent distance between the blocks. The soft normalized cut loss for K clusters is defined below:

$$\begin{aligned} soft\_n\_cut\_loss(V, w) &= \sum_{k=1}^{K} \frac{cut(A_k, V - A_k)}{assoc(A_k, V)} \\ &= K - \sum_{k=1}^{K} \frac{\sum_{u \in V} p(u{=}A_k) \sum_{v \in V} w(u,v) p(v{=}A_k)}{\sum_{u \in V} p(u{=}A_k) \sum_{t \in V} w(u,t)} \end{aligned} \tag{3}$$

The transformer include one non-linear layer with activation function Relu and a softmax layer. It constructs a [batch size $\times$ number of clusters] from the output of the encoder [batch size $\times$ latent dimension].

3.3 Evaluation Stage

In the evaluation stage, the validation data set was preprocessed to be blurred and divided into blocks. The blurred and clear block sets were then passed into the VAE with two filters and the VAE with a single decoder respectively. The resulting deblurred blocks from both networks were then reconstructed to images of the original size. Two sets of reconstruction results were compared using PSNR. We expected that a significant percent of the images reconstructed from the network with clustering and multiple filters would have higher PSNR than images reconstructed by the other network.

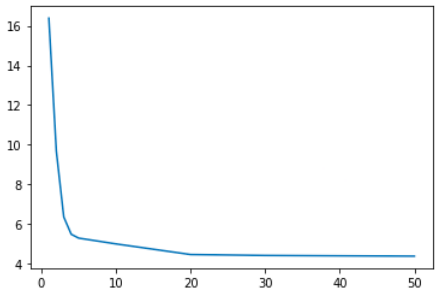
Moving forward, we could use the quality metrics in Section 2.3 to evaluate the quality of clustering. Then, compare the quality scores between single-filter autoencoder and multi-filter autoencoder.

4 Discussion

Based on the experiments, the variational autoencoder does not perform well according to the quality metrics. The convolutional autoencoder achieves much better performance according to all the metrics. In addition, the result of the experiments indicates that a multi-filter autoencoder does enhance the quality of reconstructed images.

4.1 CelebA Blocks + Variational Autoencoder

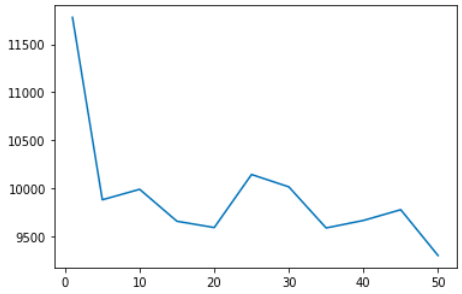
While using the variational autoencoder (Section 3.2.4) to train the blocks divided from CelebA dataset (Section 3.1.1), the outcome from the quality metric is relatively the best among all the experiments. There is a significant decrease on the training loss and the training loss converges after around 20 epochs. However, the average PSNR score of reconstructed images are around 10, which is unexpected.



A good aspect is that in 96 percent of time, the image qualities of the reconstruction of double-filter network is better than the qualities of reconstructed images from single-decoder.

4.2 CelebA Blocks + VAE with VGG16

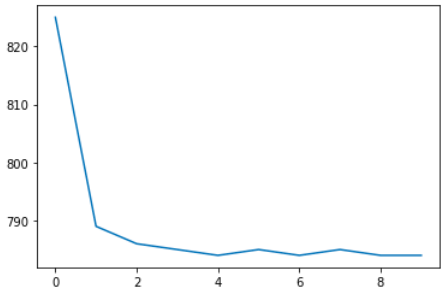
The training data and the network are referring to Section 3.1.1 and Section 3.2.2.



There is a significant decreasing on the training loss in the first 5 epochs. After 5 epochs, the training loss is fluctuating. The average PSNR score of validation set is 0.0279, which is a extremely low quality. This result indicates that VGG16 does not positively affect training procedure of the current dataset (CelebA).

4.3 CelebA Blocks + VAE with ResNet

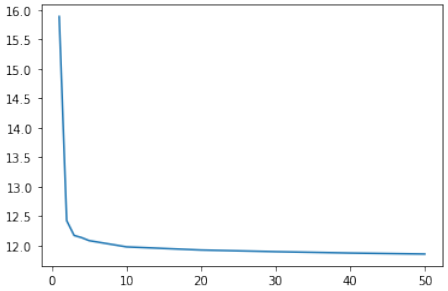
The training data and the network are referring to Section 3.1.1 and Section 3.2.3.



According to the trend of the training loss, ResNet and Basic Network are similar. However, the decreased percentage of the loss of ResNet (5 percent) is much lower than the loss of Basic Network (73 percent). This indicates the learning outcome of ResNet is not as good as the Basic Network. The quality metric PSNR score demonstrates this as well. The average PSNR score of reconstructed images is 0.0439. Therefore, ResNet does not positively impact the training process as well.

4.4 MNIST Blocks + Variational Autoencoder

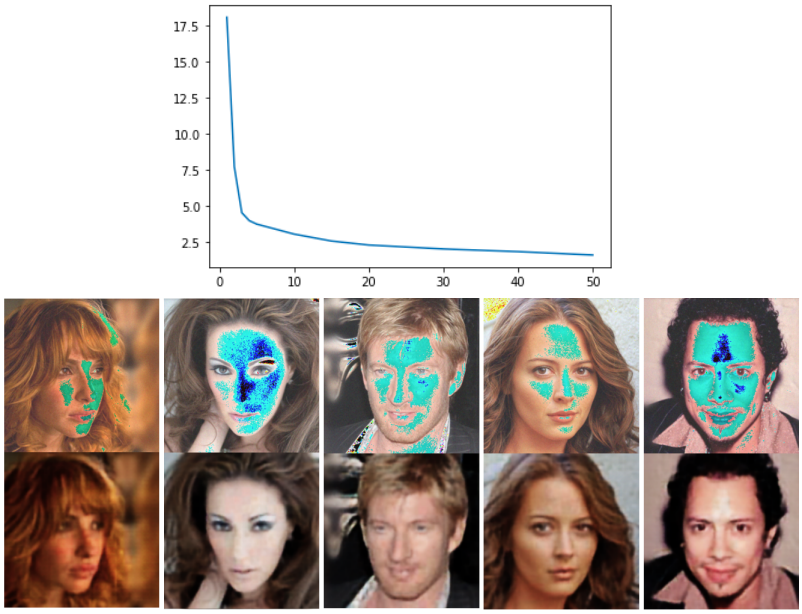
The training data and the network are referring to Section 3.1.3 and Section 3.2.1.



There is a significant decrease on the training loss, and the overall pattern of decrement is close to that in Section 4.1. The training loss converges after around 90 epochs. The average PSNR score of reconstructed images is around 10. This result shows that the basic network produce similar outcome on 1-channel images and 3-channel images.

4.5 CelebA Blocks + Basic Network

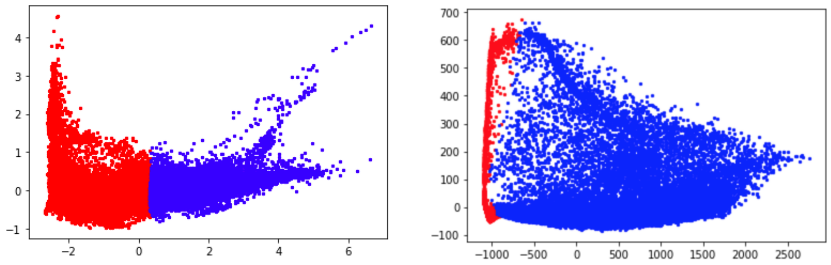
The training data and the network are referring to Section 3.1.1 and Section 3.2.1.



The Convolutional Autoencoder architecture achieves an average PSNR score of 26.7, which is a relatively good outcome base on current training-data size. Around 65% of the time, the image qualities of the reconstruction of the double-filter network are better than the qualities of reconstructed images from single-decoder. The multi-filter autoencoder improves quality scores among all the three metrics. This result indicates a multi-filter convolutional autoencoder performs better on deblurring heterogeneous blurs.

4.6 CelebA Blocks + Basic Network with Soft N-Cut Loss

The training data and the network are referring to Section 3.1.1 and Section 3.2.5. The training loss and quality scores are similar to the Basic Network (Section 4.5). The separability of the clusters is improved. Refer to the figures below, the left figure shows the clusters without using soft n-cut loss in the training process and the right figure shows the clusters with soft n-cut loss.

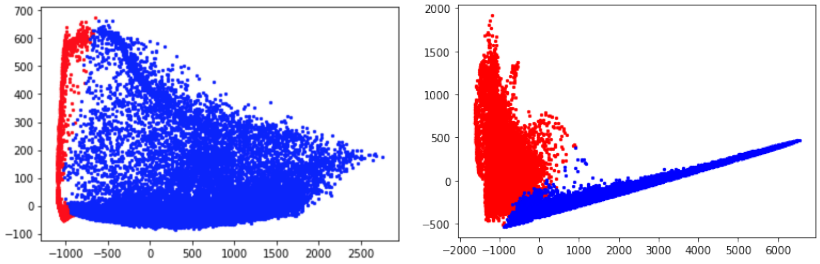


Applying soft normalized cut loss enhanced the latent space clustering. The soft-n-cut loss improved the separability of the clusters in the latent space.

4.7 CelebA Blocks + Basic Network with ResNet and Soft N-Cut Loss

Residual Network does not significantly impact both the reconstruction quality and the latent space clustering.

The latent space clustering result is shown below. The left figure is the latent space from Basic Network, the right figure is from the Residual Network.



5 Conclusion

The following table presents the all the results of the experiments using CelebA dataset. (Percentage represents the percent of time that multi-filter autoencoder gains higher quality score under a specific quality metric.)

	PSNR	Percentage	SSIM	Percentage	UQI	Percentage
Blurred/Noise	11.37		0.5593		0.5965	
Conv AE	29.07	93.33%	0.8428	100%	0.9763	94.7%
Conv AE + N Cut Loss	25.67	78.7%	0.7429	97.3%	0.9601	90.7%
ResNet + N Cut Loss	25.58	90.7%	0.7314	90.7%	0.9504	77.3%
VAE	10.37	96.3%				
VAE + ResNet	0.0439	75.3%				
VAE + VGG16	0.0279	58.7%				

References

[1] Claudio Quiros, A., Murray-Smith, R., & Yuan K. (2020). PathologyGAN: Learning deep representations of cancer tissue.*arXiv preprint arXiv:* 1907.02644.https://arxiv.org/abs/1907.02644.

[2] Ding, F., & Luo, F. (2019). Clustering by Directly Disentangling Latent Space. *arXiv preprint arXiv:*1911.05210. https://arxiv.org/abs/1911.05210.

[3] Erhan, D., Courville, A., Bengio, Y., & Vincent, P. (2010, March). Why does unsupervised pre-training help deep learning?. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (pp. 201-208).

[4] Gong, D., Yang, J., Zhang, Y., Reid I., Shen C., van den Hengel, A., & Shi, Q. (2016). From Motion Blur to Motion Flow: a Deep Learning Solution for Removing Heterogeneous Motion Blur. *arXiv preprint arXiv:* 1612.02583. https://arxiv.org/abs/1612.02583.

[5] He, K., Zhang, X., Ren, S., & Sun, J(2015). Deep residual learning for image recognition. *arXiv preprint arXiv:* 1512.03385. https://arxiv.org/abs/1512.03385.

[6] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167. https://arxiv.org/abs/1502.03167.

[8] Kingma, D. P., & Welling, M(2013). Auto-encoding variational bayes. *arXiv preprint arXiv:* 1312.6114. https://arxiv.org/abs/1312.6114.

[9] Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative Flow with Invertible 1×1 Convolutions. *arXiv preprint arXiv:* 1807.03039. https://arxiv.org/abs/1807.03039.

[10] Xianxu Hou, Linlin Shen, Ke Sun, Guoping Qiu. (2016). Deep Feature Consistent Variational Autoencoder. *arXiv preprint arXiv:* arXiv:1610.00291. https://arxiv.org/abs/1610.00291

[11] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2016). "Deep Residual Learning for Image Recognition". Proc. Computer Vision and Pattern Recognition (CVPR), IEEE. Retrieved 2020-04-23.

[12] Xide Xia, Brian Kulis. (2017). W-Net: A Deep Model for Fully Unsupervised Image Segmentation. *arXiv preprint arXiv:* arXiv:1711.08506. https://arxiv.org/abs/1711.08506