Variational auto-encoders and GMM

Badrinath Singhal (23302), Gaurav Talekar (21030)

September 2023

We made GMM with 3 different components of gaussian. 3 components are 5,10,15. We first resized the animal data to 28×28 and loaded on an numpy array using opencv library. The GMM iteration for 3 components converged. The generated images for those GMMs are seen in fig 1

The log likelihood plots for EM algorithms are in fig 2. We notice an interesting data, GMM iterations are significantly low for gaussian with 5 and 15 components whereas gaussian with 10 components converges after many iterations. This can signify that 10 gaussians are a bad fit compared to other two GMMs. This also is supported by the generated samples in fig 1

Normalised mutual information (NMI) score is for GMM with 5 gaussian is 0.0144, whereas NMI for GMM with 10 gaussian is 0.0424. For GMM with gaussian 15 NMI is 0.0344.

We used VAE to be a CNN model with fully connected layer towards the end as encoder and the opposite of that as decoder. We set the latent dimension to be 64. The reconstruction picture can be seen in fig 3. The 10×10 plot of reconstruction is seen in fig 5 whereas its original can be seen in 4. We've also generated 10×10 plot for new samples generated images which can be reffered to fig 6

The training data was same for all the questions in this assignment so we used same dataloader for this as well as rest of the questions. Fig 8 shows the combined loss plot along with KL loss and likelihood loss.

We built a classifier based on CNN for classifying images in animals dataset, the network consists of two conv layer with maxpool with 2 fully connected layer using relu as non linearity. We used cross entropy loss for training and trained it for 10 epochs. The loss and accuracy graph can be seen in fig 9

Similarly we trained MLP classifier for latent vectors produced using trained VAE. The MLP consist of 4 fully connected layers with relu as non linearity. For both the classifiers we used Adam optimizer with learning rate of $1e-3$. We achieved validation accuracy of almost 80%. The loss curve can be seen in fig **??**

We noticed that classification performance using original data outperforms classification using latent vector by a huge margin. Although interesting thing to notice is that the classification performance of latent vector classification model is better than a random classifier (which would provide accuracy of 50%). This shows that the latent vector sampled from their Gaussian distribution encodes some information of the original class on the basis of which the classifier can differentiate between classes better than a random classifier. The classifier trained on latent vectors uses fewer epochs as well as fewer parameters compared to CNN based classifier network. Using such classifier is beneficial when we have limited resources and still want decent performance. Although such methods leads to poorer performance and thus such methods can be thought of as a trade-off between resources and performance hence usage of one vs another will depend on each situation.

For beta VAE we used the same model architecture as previous question as well as the same training environment. We change value of beta in our loss fuction which was originally consiedered as 1. We took 4 different beta values which are 0.1, 0.5, 1.5, 2.0. We took latent dimension here to be 128 and trained the beta vae for each of them and generated the image for each beta vae.

For beta value 0.1 and 0.5 generated image can be seen in fig 11 and for beta 1.5 and 2.0 generated image are in fig 12.

We have interpolated the latent vectors as 10 steps between two latent vectors from two randomly sampled images. We did this 10 times and thus we have 10 pair of images and interpolated 10 latent vectors from each pair and generated the images for each interpolation.

We can see that interpolated latent space generates images which looks like the image is transitioning from $1st$ image to $2nd$ image in every interpolation. This gives indication that latent dimension is continuous and each vector have the corresponding image which is similar to the closest latent vector of one of the images in dataset.

We notice an interesting observation related to beta value. As beta value decreases we get better generation of images which is opposite of what was discussed in class. This property needs to be further investigated and analysed.

A vanilla auto encoder with three hidden linear layer on both encoder and decoder was chosen for this task. Hidden layer have 512, 64 and 64 neurons neurons whereas dimension of latent vector is 32.

The training loss of autoencoder was dont with SGD optimizer and MSE loss with learning rate of $1e-3$. May iterataions of the training was done to see the results fig 14 shows the training loss for our network. The graph of loss shows the training loss consistently going down over epochs which signifies that model may have learned the required representations.

We stored the latent vectors for all the training and validation data as another tensor file along with the label for training a classifier. We built a classifier with stored latent vectors. The classification network consist of 4 hidden layers with 128 neurons each and 3 classes. We used SGD optimizer with learning rate of $1e-3$ and cross entropy loss. Fig 15 shows the classification loss for 20 epochs. The model gave validation accuracy of 66%.

The classifier training achieves a validation accuracy of 54% which is just better than a random classifier but still not good enough like VAE. This signals that auto encoder latent representation is losing important information in encoding process, to tackle that we can increase the dimension of latent vector initially. Compared to this, the classification accuracy for latent vectors in VAE was 80%

We also reconstructed the images using the trained auto encoder with validation data. The reconstruction can be seen in fig 16. We can see that the reconstructed image is blurred significantly but we can still infer that the reconstructed image represents the original image in most of the cases.

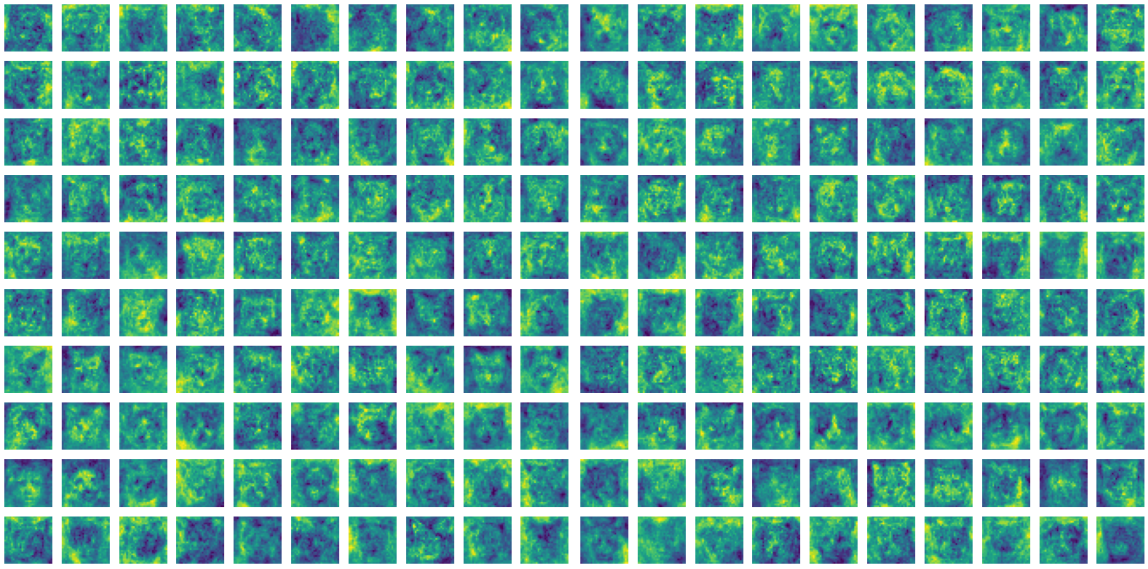
To fit GMM on the latent vectors we use sklearn's inbuilt library for that. We used 10 gaussian component to fit GMM with. It took 46 iterations for GMM to converge with final log likelihood to be 52.267.

We randomly sampled 100 latent vectors from fitted GMM and those latent vectors are inferred on trained decoder of autoencoder. Those images are plotted in 10×10 grid. Fig 17 shows the grid of image.

The generated image with help of GMM and auto encoder generated images mostly from one class, but the thing to notice is that the latent space in auto encoder signifies some image in data space and thus can also work as a generative model. Although the images are quite blurred in and represents images mostly from class 'cats'.

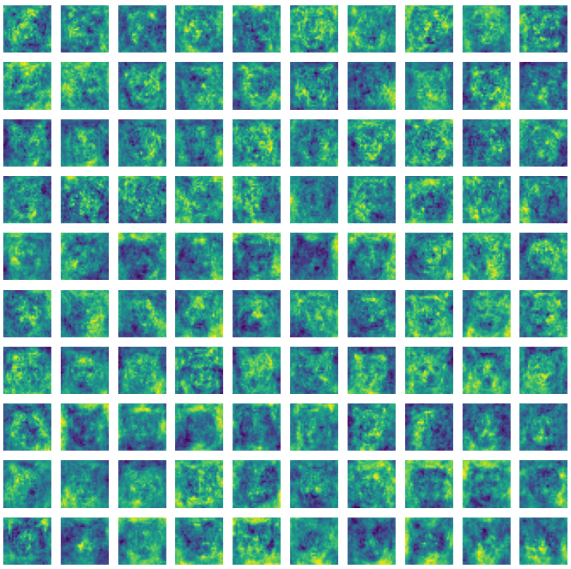
Although VAE are studied as generative models but it was suprising to see that a normal auto encoder with GMM sampled latent vector can act as a generative model too. The quality of GMM

generated sample in question 1 is comparable to that of generated samples in auto encoder, although GMM in question 1 generates wider variety of images than auto encoder. Assesing the qualities would be a difficult task as the GMM images in question 1 is of dimension 28×28 but in auto encoder it is 128×128. Since auto encoder images are of higher resolution it seems like the images generated by auto encoder is better in quality than that of GMMs in question 1. But this should be further investigated thoroughly.



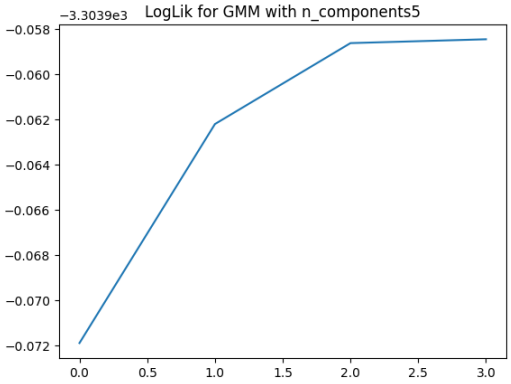
(a) GMM with 5 gaussians

(b) GMM with 10 gaussians

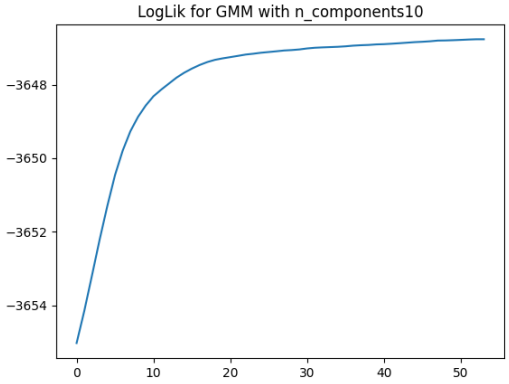


(c) GMM with 15 gaussians

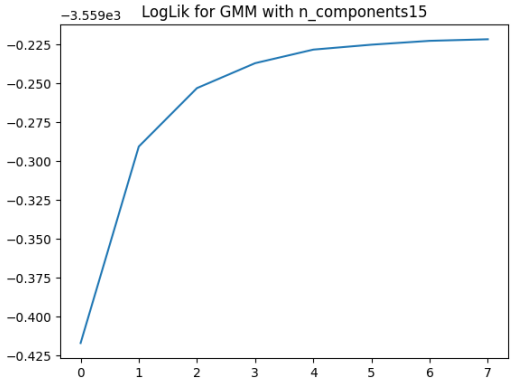
Figure 1: GMM generated samples



(a) GMM with 5 gaussians



(b) GMM with 10 gaussians



(c) GMM with 15 gaussians

Figure 2: GMM Log Likelihood plot

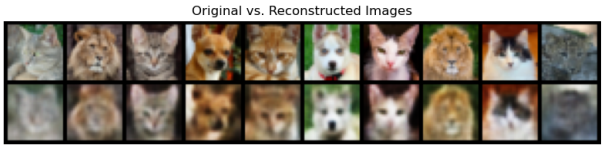


Figure 3: Reconstruction vs Original images in VAE

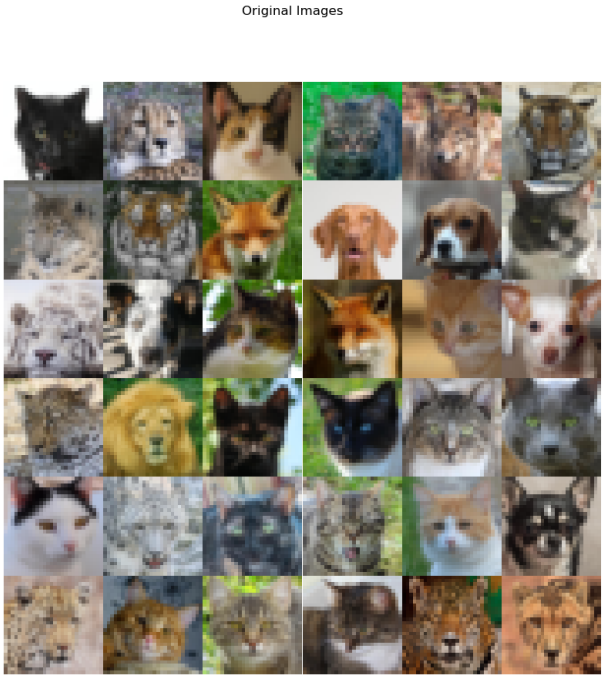


Figure 4: 10x10 plot for original

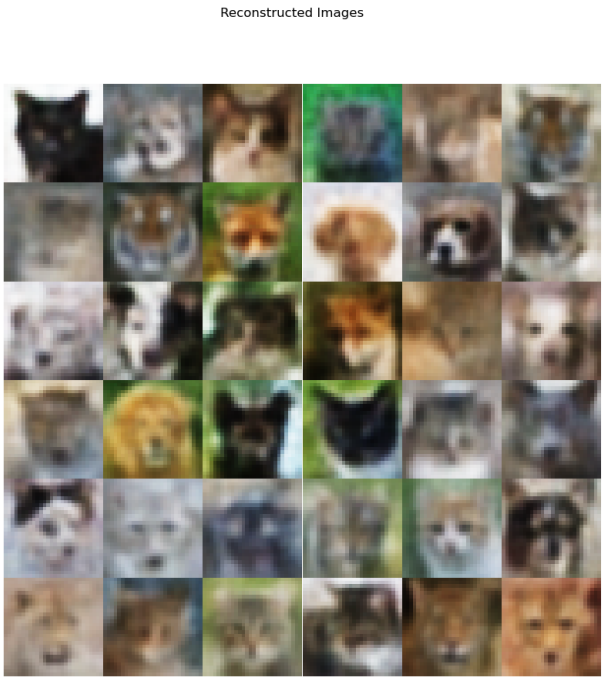


Figure 5: 10x10 plot for reconstruction

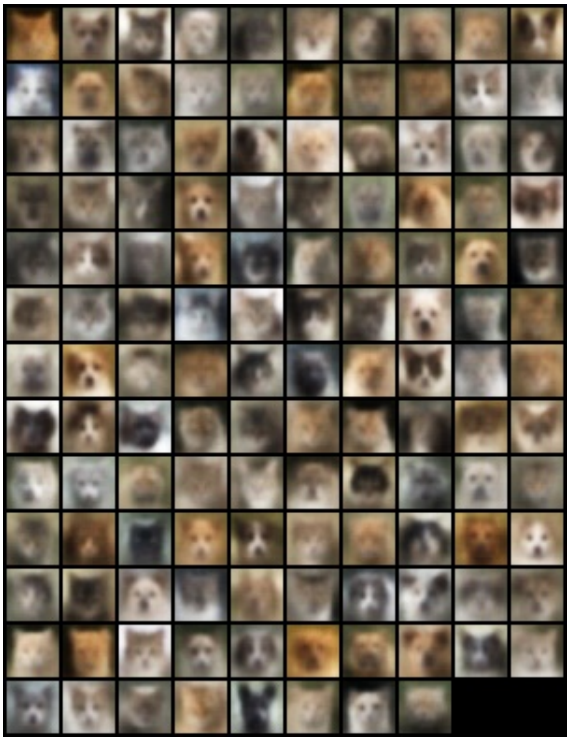


Figure 6: 10x10 plot for generation

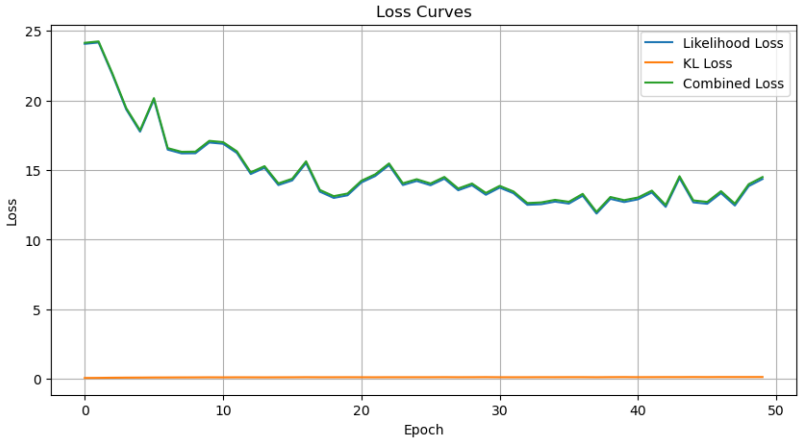
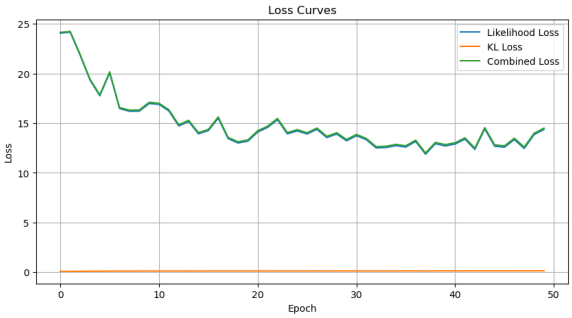
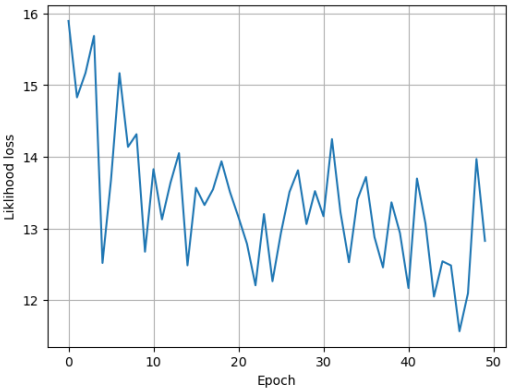


Figure 7: Loss for VAE

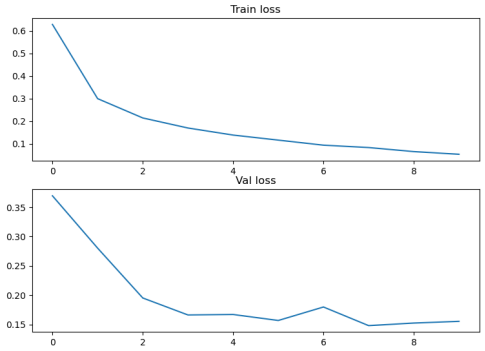


(a) KL and combined loss VAE

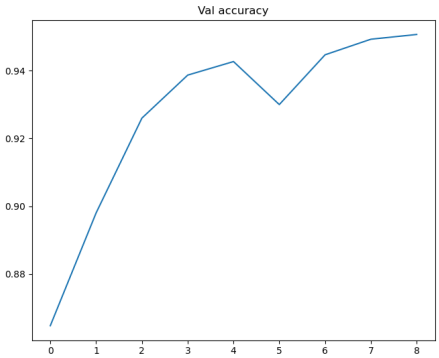


(b) Likelihood loss for VAE

Figure 8: Plot of loss over epoch

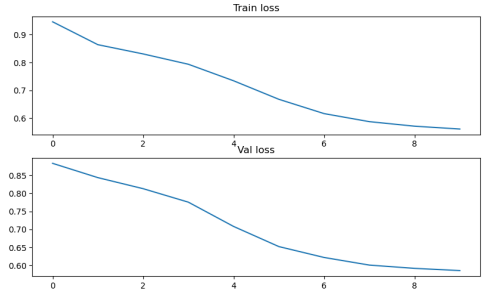


(a) Train and Val loss for classification

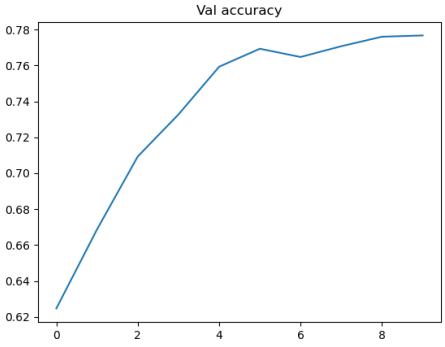


(b) Validation accuracy

Figure 9: Loss and accuracy plot for CNN based classification on animals dataset

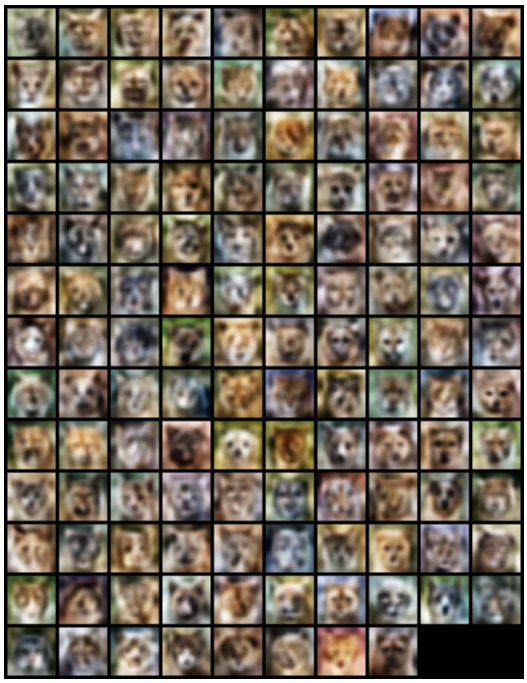


(a) Train and Val loss for latent vector classification

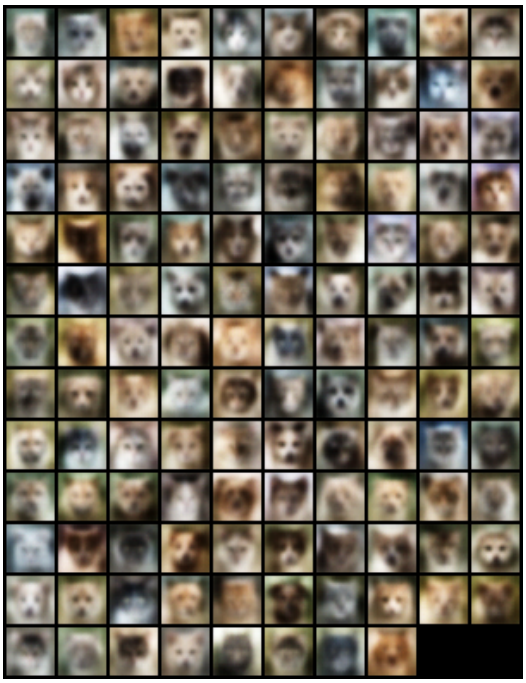


(b) Validation accuracy for latent classification

Figure 10: Loss and accuracy plot for latent vector classification on animals dataset

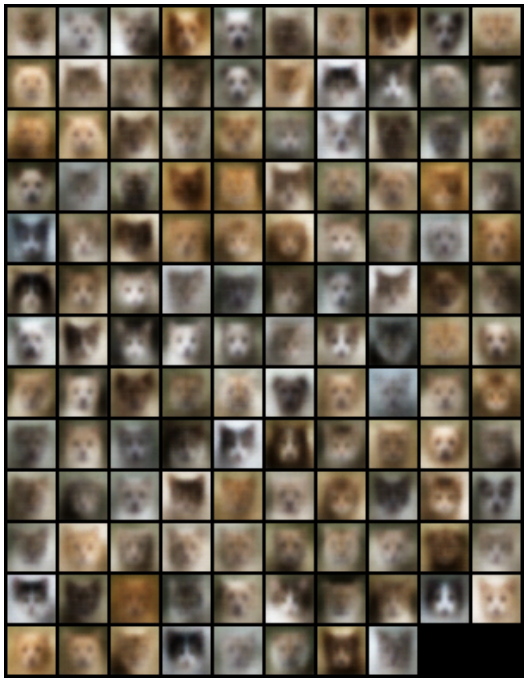


(a) Generated images for beta 0.1

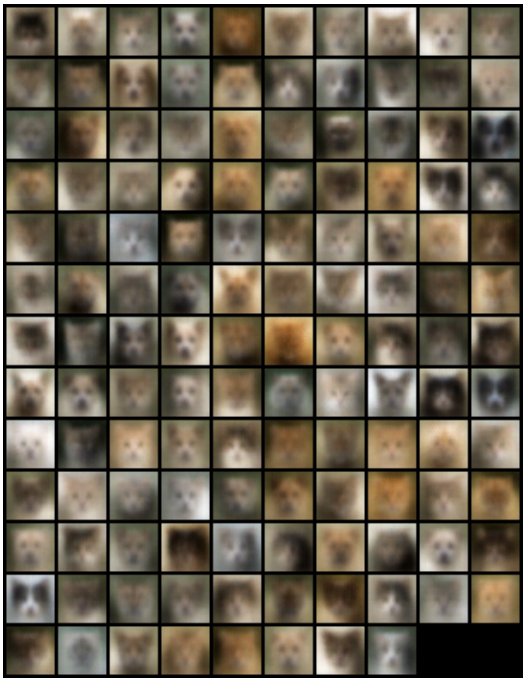


(b) Generated images for beta 0.5

Figure 11: Generated image for beta 0.1 and 0.5



(a) Generated images for beta 1.5



(b) Generated images for beta 2.0

Figure 12: Generated image for beta 1.5 and 2.0

(a) Interpolated image for 1st pair of image

(b) Interpolated image for 2nd pair of image

(c) Interpolated image for 3rd pair of image

(d) Interpolated image for 4th pair of image

(e) Interpolated image for 5th pair of image

(f) Interpolated image for 6th pair of image

(g) Interpolated image for 7th pair of image

(h) Interpolated image for 8th pair of image

(i) Interpolated image for 9th pair of image

(j) Interpolated image for 10th pair of image

Figure 13: Interpolated images between two images

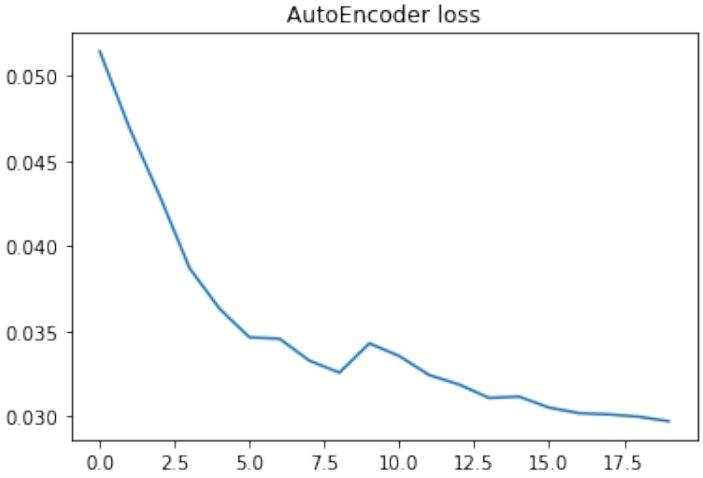


Figure 14: Autoencoder training loss

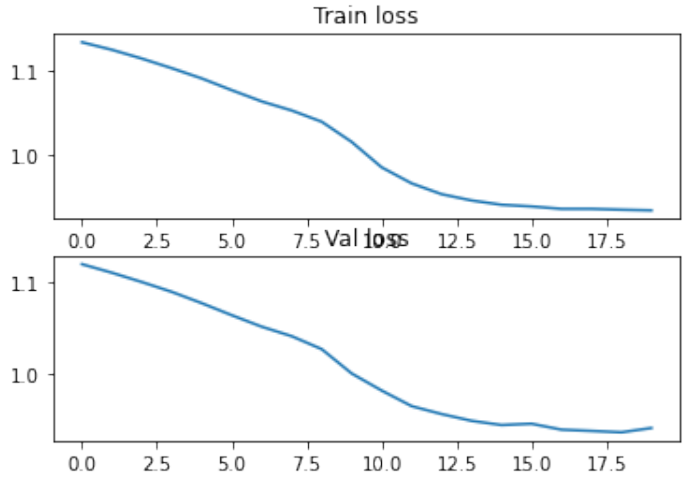


Figure 15: Autoencoder classification loss

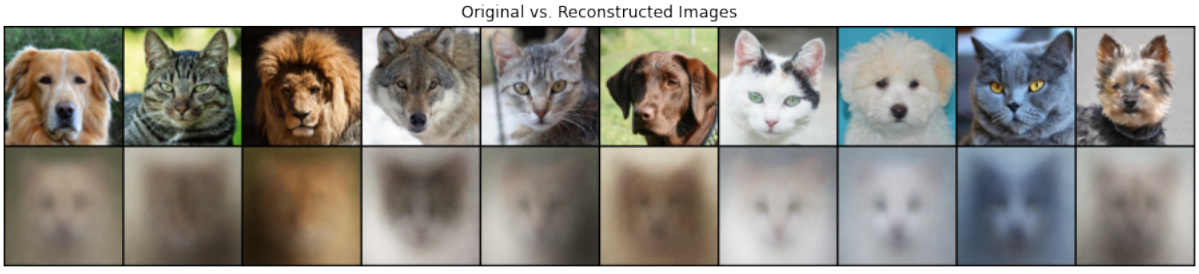


Figure 16: Autoencoder reconstructed images

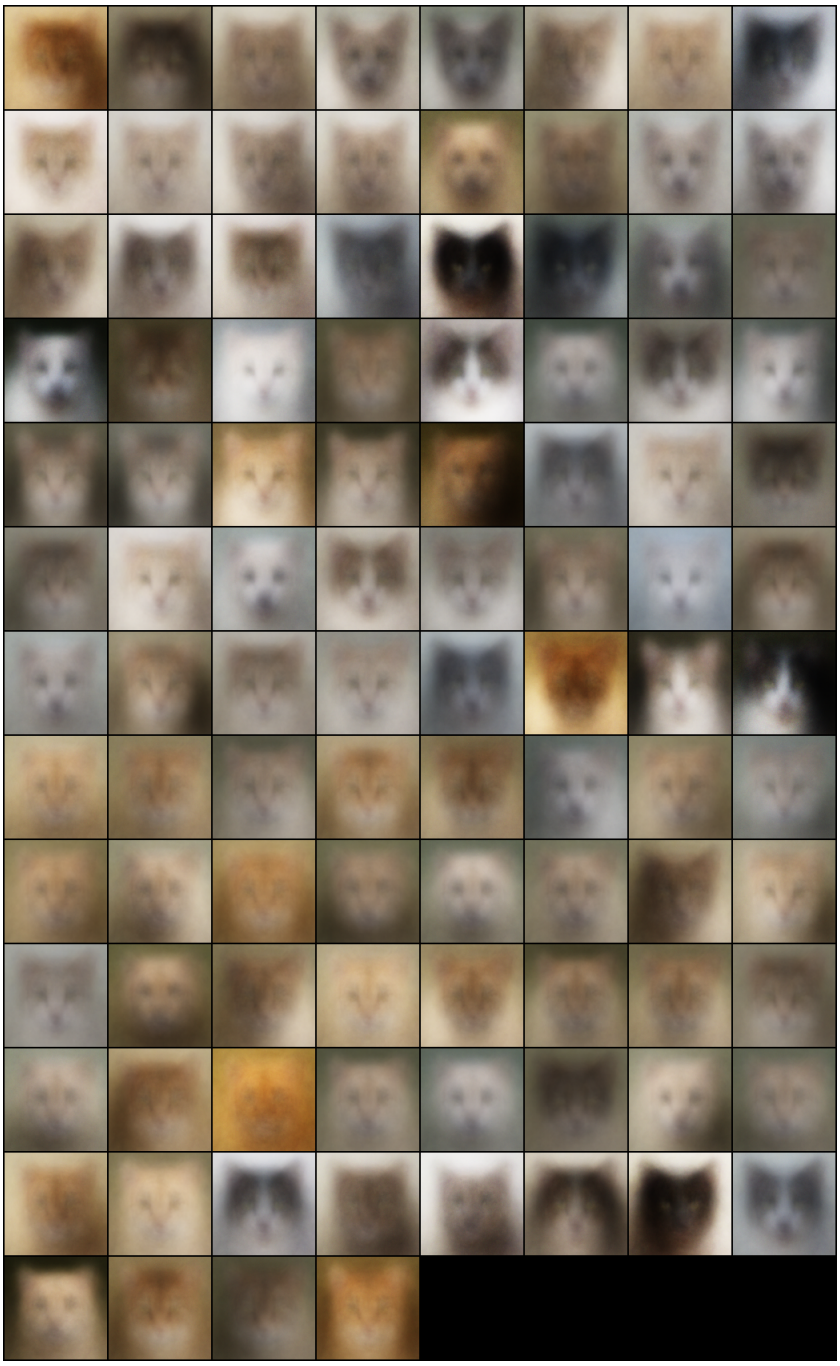


Figure 17: New sampled images using GMM generated latent vector