Assignment 3
Pattern Recognition and Neural Network

Alokendu Mazumder (20134)* , Manan Tayal (20133)**, Piyush Tiwary (19897)†, and Saurabh Shrivastava (19881)‡

Email: {*alokendum, **manantayal, †piyushtiwary, ‡saurabhshriv}@iisc.ac.in

***Abstract*—We perform several experiments on tiny image net data set using the algorithms taught in lecture. Specifically, in this assignment, we have implemented Generative models from scratch and tested on relevant data sets.**

I. Introduction

In this assignment we cover experiments/investigations on GANs, VAE, t-SNE and compared the above plots using PCA on the same data set . The reference codes (where-ever needed) is tagged.

II. DCGAN

We have implemented DCGAN over tiny-imagenet data-set. For training of GAN, 19k samples are used from test/validation folder of tiny-imagenet and rest 1k is used for testing purpose. Each image is resized to $(56, 56, 3)$ to feed it as input of discriminator. The generator samples noise from a zero mean identity covariance Gaussian distribution of 100 dimensions.
Note, that during passing both the batch of generated images and test images via FID module, we up-sampled them to $(75, 75, 3)$ as inception module required inputs of this dimension at least.
The results we got, the generated images are very bad. The generator has not learnt well. I can infer reasons of the failure by looking at the generated images, which are as follows:

- In the grid, we can see a same image is being repeated huge number of times. Hence, the generator is generating same ouput for different noise samples. It's called mode collapse. But this mode collapse is different from what we usually read in architecture. This is happened due to the fact that discriminator is learning slower than the generator, with discriminator being bad, generator picks up a random bad image which confuses the discriminator.
- The learning of discriminator and generator are not tuned together probably.
- Due to less number of epochs to fit model. (It takes long time to train a GAN)

We evaluated our results with respect to FID scores as shown in Table I.
Figure 1 has GAN generated images.

III. Variational Auto-Encoder (VAE)

We have implemented VAE over tiny-imagenet data-set. For training of VAE, 19k samples are used from test/validation folder of tiny-imagenet and rest 1k is used for testing purpose. Each image is resized to $(28, 28, 1)$ to feed it as input of encoder. The decoder samples from latent variable of dimension ($\mathbf{k}$ = 2, fixed).
Note, that during passing both the batch of generated images and test images via FID module, we up-sampled them to $(75, 75, 1)$ and converted them into three channel RGB image, as inception module required inputs of this dimension at least.
The results generated are not upto the mark, but from the generated images we can infer about what lead to bad trainig, as follows:

-

We evaluated our results with respect to FID scores as shown in Table I.
Figure 2 has VAE generated images.

IV. Dimensionality Reduction using PCA and t-SNE

Performed PCA on the bloodmnist dataset for the top 50 principal components. Plotted the graphs in 2D and 3D for the top 2 and 3 components. Also then reconstructed one of the images back using the top 20 principal components (this we did for the grayscaled version of the particular image). Plotted both the original and reconstructed images. For tSNE, we first performed PCA on the bloodmnist dataset for 50 principal components and then applied tSNE on the top of it. Plotted the corresponding graphs in both 2D and 3D for the top 2 and 3 principal components. For both PCA and tSNE, the 2D and 3D plots are corresponding to the training dataset which contain 11959 images.We can clearly see that tSNE is way better in terms of visualisation than PCA.

Model	DCGAN	VAE
FID (in 10^4)	40	51

TABLE I: DCGAN VAE

Fig. 1: GAN generated images

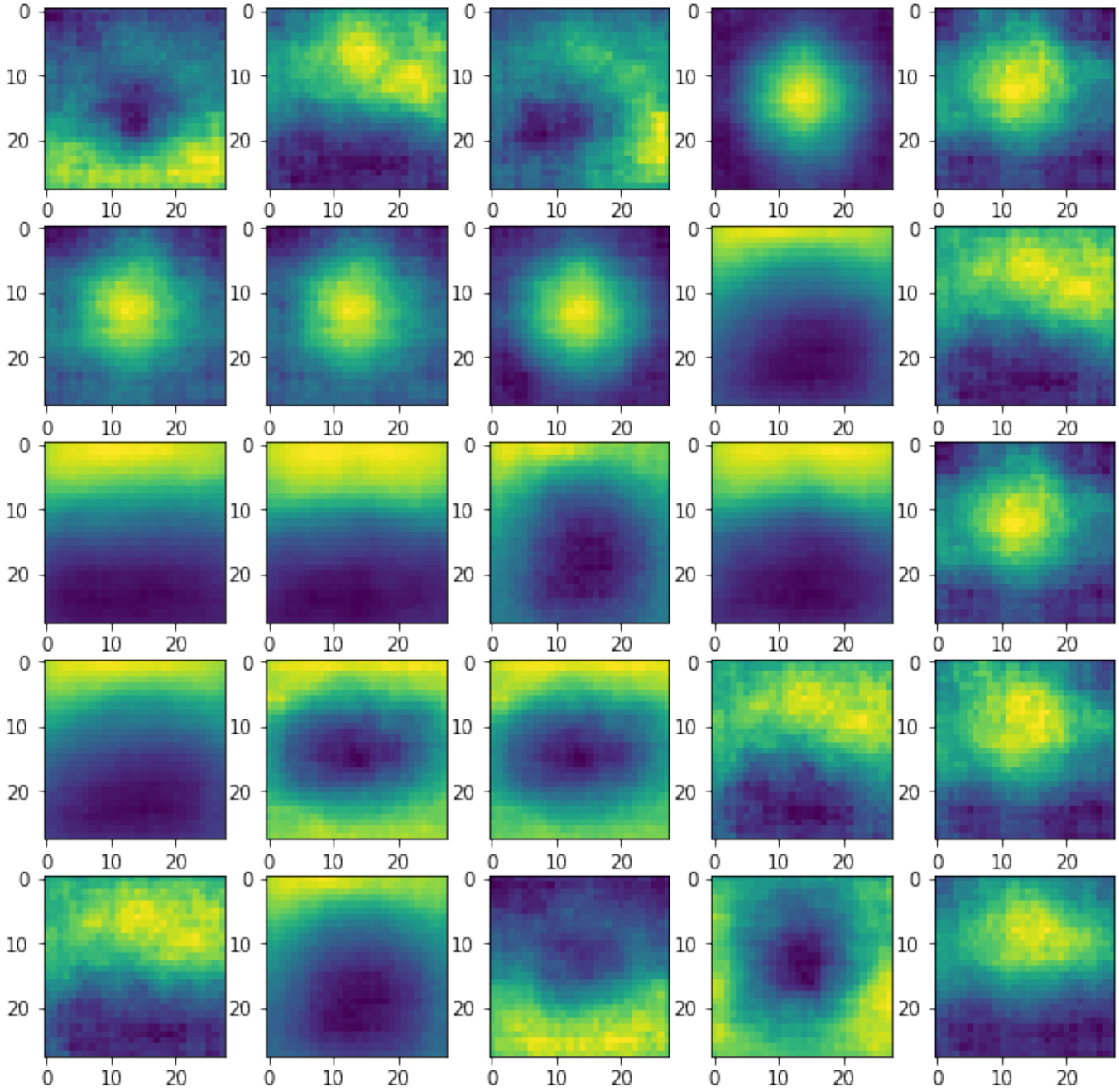
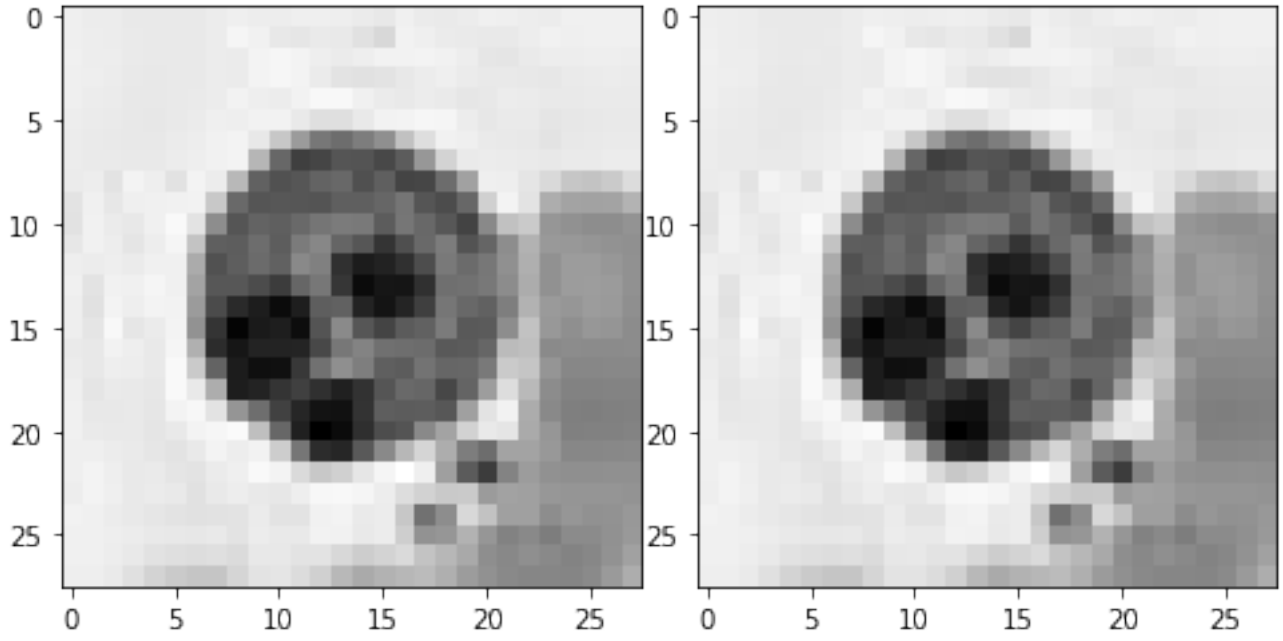
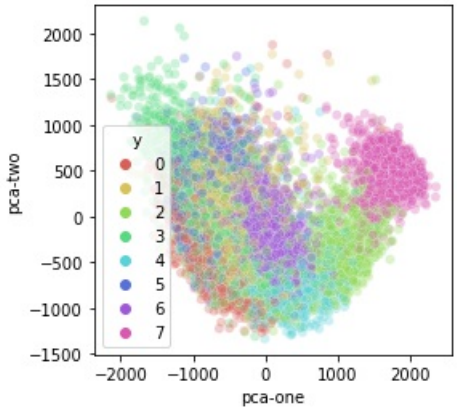


Fig. 2: VAE generated images for **k** = 2

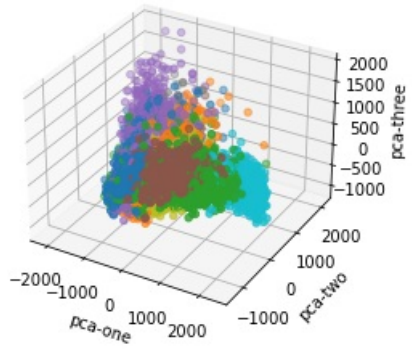


(a) Original Image

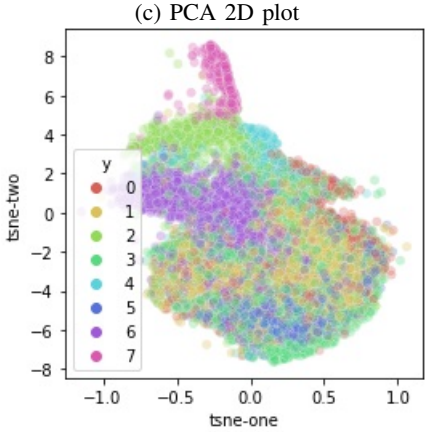
(b) Reconstruction Image



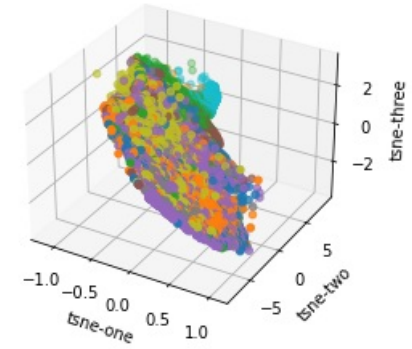
(c) PCA 2D plot



(d) PCA 3D plot



(e) tSNE 2D



(f) tSNE 3D

Fig. 3: Plots of PCA and tSNE