DL End Sem Assignment

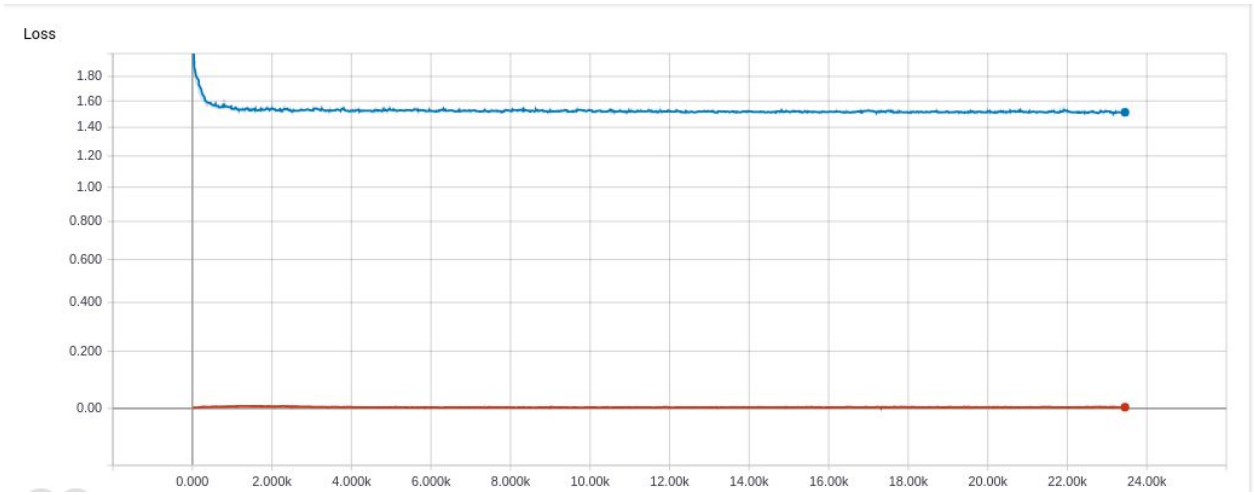
Siddharth Yadav | 2016268

Question 1

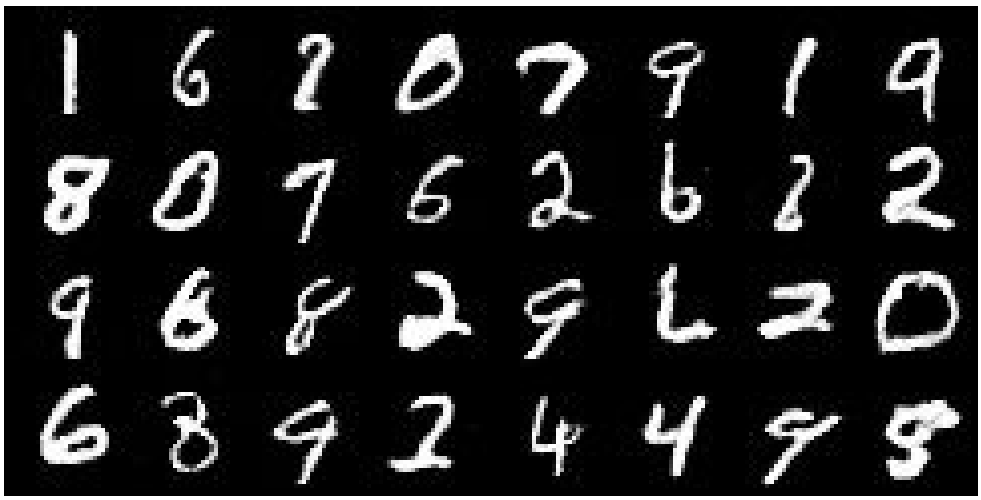
m=16

Loss Curve(m=16)

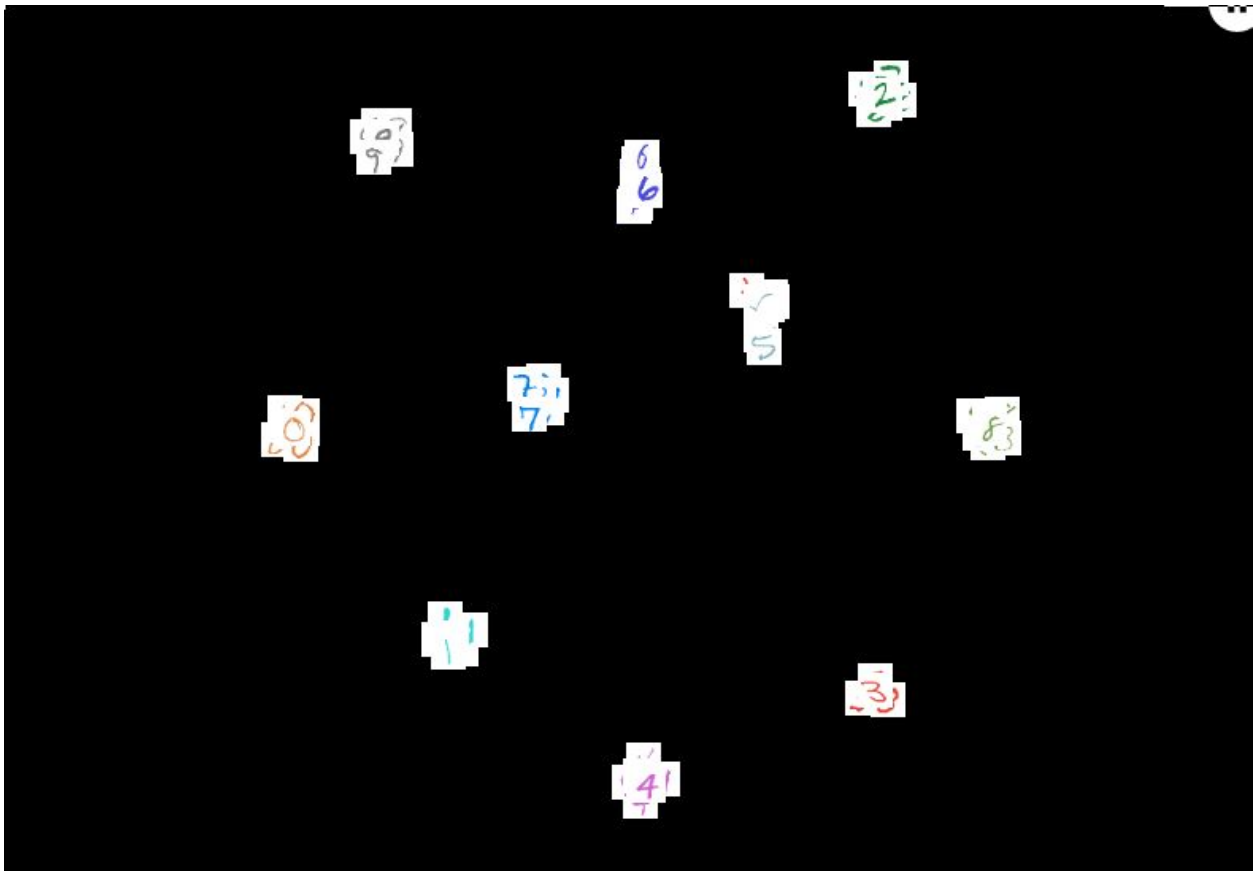
Blue is for discriminator loss; Red is for generator loss



Generated Images(m=16)

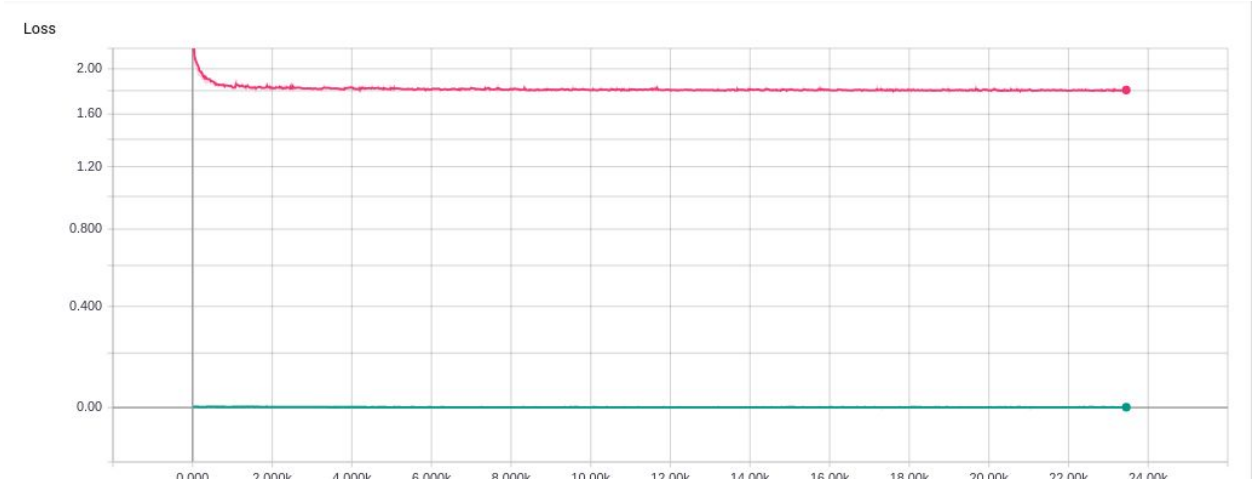


T-SNE of features (m=16)

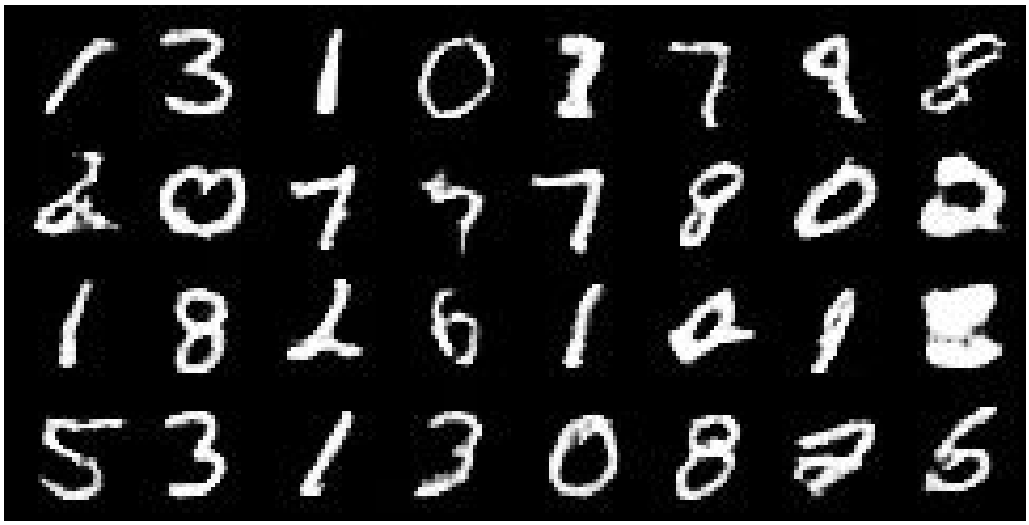


M=32

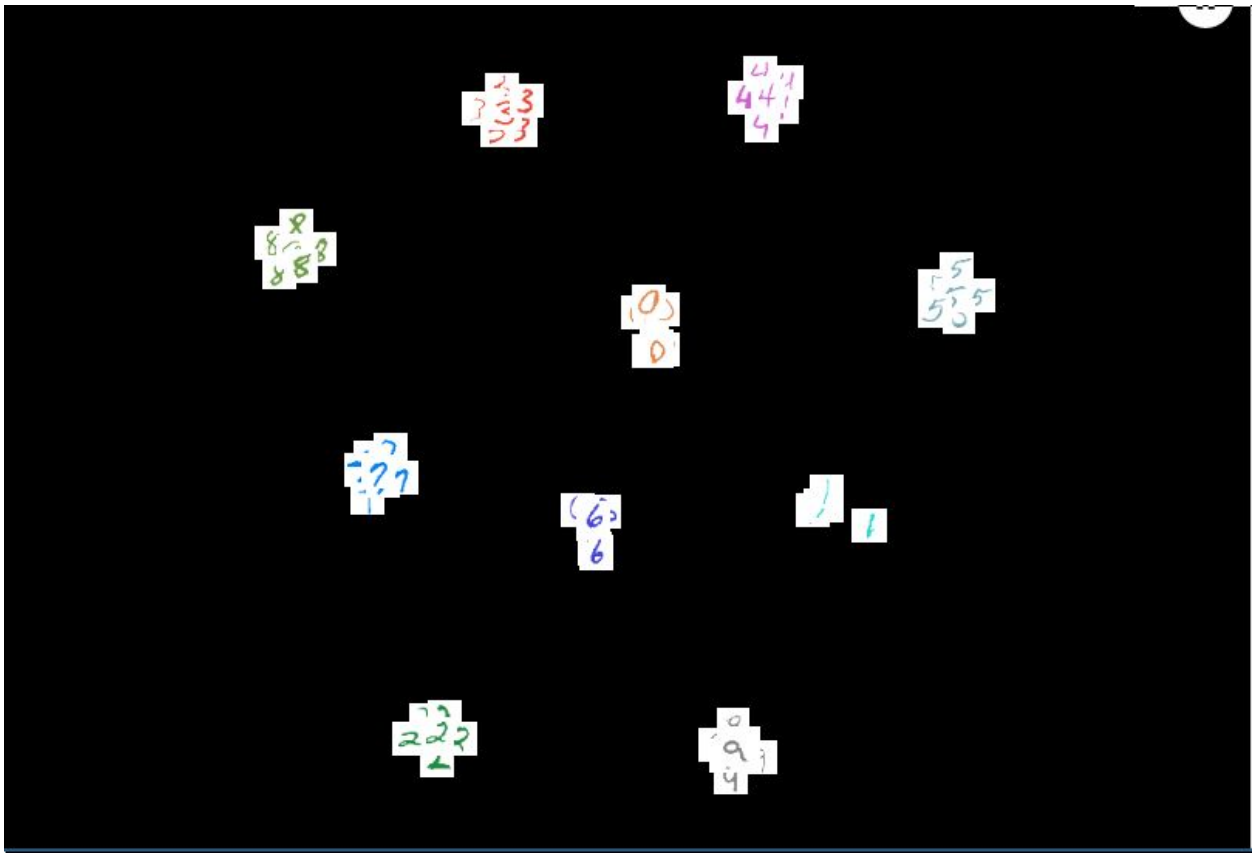
Loss Curve (m=32)



Generated Images (m=32)



T-SNE of features (m=32)



Note:

- Each model has been trained for 50 epochs where each epoch had exactly 60,000 randomly selected triplets

Table 3: Revisited

(9-NN for N=100)	m=16	m=32
Accuracy	0.989	0.9903
mAP	0.99151	0.99253

Table 4: Revisited

(9-NN for m=16)	N=100	N=200
Accuracy	0.9890	0.9898
mAP	0.99151	0.99178

The accuracy I found is considerably higher than what was reported in the paper. A possible reason for it might be that they trained there GAN only use 100 labeled examples from the training set whereas I used all of the training examples.
But the good thing is the accuracy follows the intuition of increasing when we go from m=16 to m=32 and from N=100 to N=200.

Things tried

Even after trying a lot of hyperparameter tuning and ways of training a vanilla GAN(with linear layers), I wasn't able to get the model to train properly. What a tried with Vanilla GAN:
- adding normal noise to the input of discriminator
- making positive labels between 1.2 and 0.8 and negative labels between 0.0 and 0.3
- BatchNorm
- Dropout
- LeakyReLU
- weight normalization of the last layer of the generator and all layers of the discriminator
- Normalization of image inputs
- Increasing and decreasing number of units in linear layers
- SGD for Discriminator and different learning rate

Hence, I shifted to DCGAN!

With DCGAN, I implemented all of the above things by default and I tried the following loss functions while training the TripletFAN
- Triplet Margin Loss + HingeLoss + Hinge Loss: didn't work
- Triplet Margin Loss + HingeLoss + feature matching loss: didn't work
- Triplet Margin Loss + f_discriminator_unsupervised_loss + feature matching: worked nicely
- Triplet Loss as in paper + f_discriminator_unsupervised_loss + feature matching loss: worked nicely

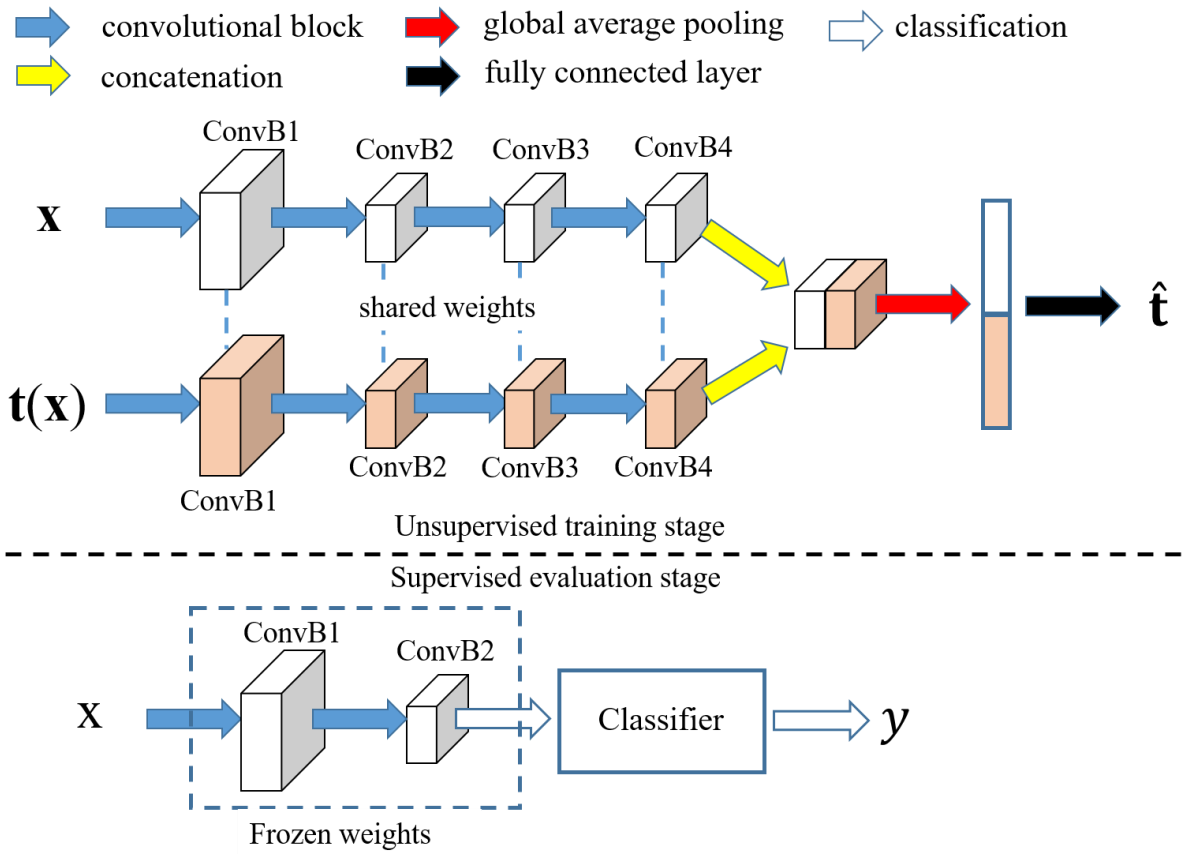
References:

- For DCGAN (the basic template of this code was taken from here)
  - https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
- For various hacks for gan models and trainings:
  - https://github.com/soumith/ganhacks/
  - https://github.com/Sleepychord/ImprovedGAN-pytorch
- For details related to the paper:
  - https://github.com/maciejzieba/tripletGAN
  - https://arxiv.org/pdf/1704.02227.pdf

Question 2

Thanks to, the capacity of deep learning models to learn high-quality semantic features, it has been successfully applied to tasks like object detection, segmentation, autonomous driving car, etc. Though, its requirement of a large amount of data makes it infeasible to apply it to new problems. Unsupervised methods for learning semantic features are important to successfully apply deep learning to problems without much data. A few popular unsupervised methods include autoencoding, VAE, GAN, and self-learning methods. Existing auto-encoders have an encoder that learns to represents the input in low dimensional space for reconstructing back the input using a decoder. This paper introduces Auto-Encoding Transformations(AET), a new way of unsupervised representation learning. It has one encoder that learns to represent both the input image/data and its transformation in low dimensional space and a decoder which reconstructs the corresponding transformation operation. Therefore, the loss functions for training depends on the difference between predicted transformation and the original transformation.

Many kinds of transformations can be used for representation learning in AET. Parameterized transformation: where a set of parameters, like a rotation or translation matrix, mathematically defines the transformation. GAN-induced transformation: where a generator learns to produce a transformed image for a particular image as a function of a noise parameter z. AET models try to predict the noise z from the given original and transformed image. Non-parameterized transformation: when it is not possible to represent the transformation with a fixed set of parameters, it still can also be used for training AET by using the approximate difference between the transformations of the randomly sampled image as the transformation parameters. The author has used parametric transformation for his experiments because it doesn't involve training another model like GANs and it gives better results than the non-parametric counterparts. In the parametric transformation, only affine transformations, which is a composition of a random rotation, translation scaling, and sheering; and projective transformations, formed by random translation of the four corners of the image, scaling, and rotation; are considered.



For evaluation purposes, the AET is first trained with the respective dataset images. AETs encoder is a Convolution Neural Network with 4 to 6 convolution blocks. The transformeand image and the original image share this encoder. The concatenation of features representation of both the image is given to the decoder, which is a fully connected layer. After training this set of encoder and decoder in for predicting the transformation parameter, we add a classification block on top of the feature representation obtained from the encoder. The paper has shown a significant increase boost for CIFAR-10, ImageNet and Places datasets when compared to the previous state-of-the-art. The exact configuration of AETs and classifier architecture depend on the dataset used and the most popular configuration for evaluation for that dataset.

The paper shows a very detailed and fair quantitative comparison of AET with respect to a wide range of existing techniques by making the setup very similar to the popular evaluation setups for the particular dataset. It's also the first paper to use affine and projective transformations as a learning objective. The paper lacks a qualitative analysis of the proposed model and a discussion on the reasons for its better performance. It's a decent paper introducing a simple idea with proper evaluation and experiments.