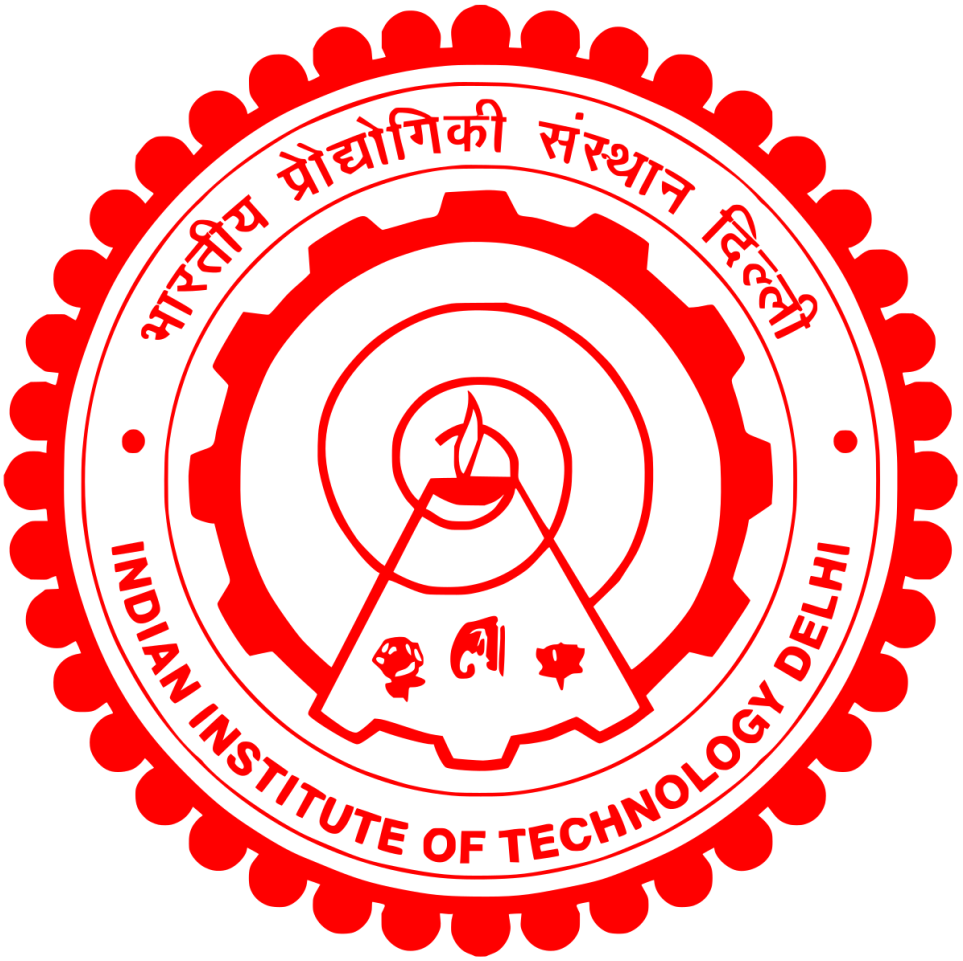
COL870: Assignment 2 | Report

Ankit Garg 2017EE10439
Devang Mahesh 2017EE10093

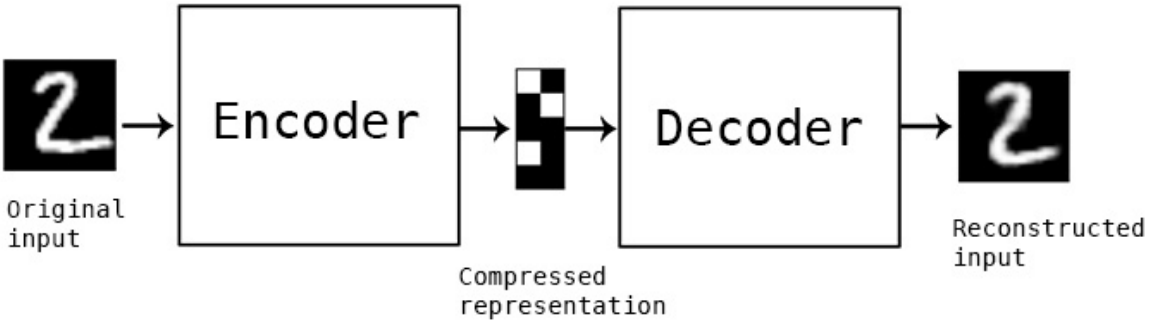
Link to the Model Weight Folder:
https://drive.google.com/drive/folders/1XmG9sZecJXI0ZbvEgJv50oHjMGDrQjVM?usp=sharing

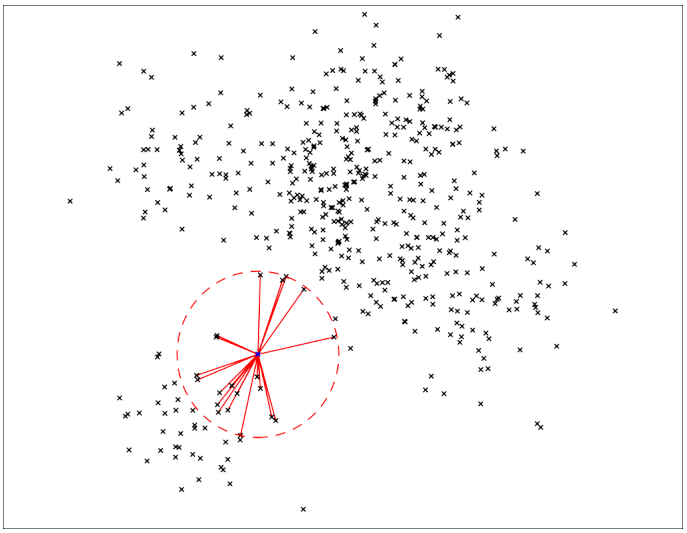


Part 1
Semi Supervised Labelling

In this part, we focus on labelling the query and target digits using the 10 labels provided already. The pipeline that we follow for this is:

- Using a convolutional autoencoder to learn low dimensional and relevant features from the input image
- After obtaining the features of the query and labelled images from the trained autoencoder, we use the nearest neighbour algorithm to label the unlabelled data. The reference point the are features of the labelled data and then all the other features are labelled wrt to them



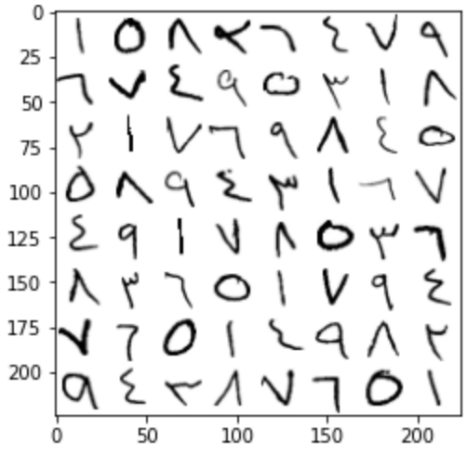


(Image Ref: Building Autoencoders in Keras + Nearest neighbor methods and vector models – part 1 · Erik Bernhardsson)

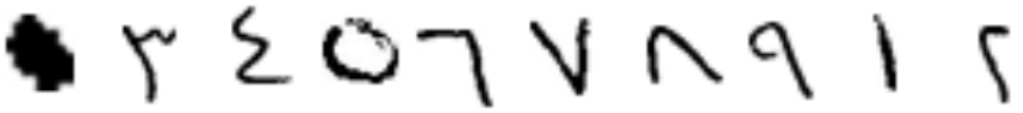
Loss Function for AE	Pixel wise weighted binary cross entropy
Regularisation on latent	The hidden representation of the labelled images

variable	should be far in low dimensional space. Cosine similarity was used
Number of epochs	60
Optimiser + lr	Adam; lr = 1e-2 with Multiplicative LR:0.95

Labelled sudoku image, using the above method :-




```
Label
[8, 7, 6, 7, 4, 2, 5, 7]
[4, 2, 2, 7, 3, 8, 8, 6]
[1, 8, 5, 4, 7, 7, 2, 6]
[3, 6, 7, 2, 1, 8, 7, 5]
[2, 5, 8, 5, 2, 3, 2, 4]
[2, 1, 4, 3, 8, 1, 2, 2]
[5, 8, 7, 8, 2, 7, 7, 8]
[7, 2, 8, 4, 5, 2, 3, 8]
```



In the above labelled target image, we see that the labelling done is quite good considering we had only 9 labelled images to start with. Still there is noise present and the output is not a valid sudoku board. The labels 3 and 7 are confused with one another. It means that in the latent representation they are quite close. We continued with these labels to train the GAN.

Part 2
Conditional Gan

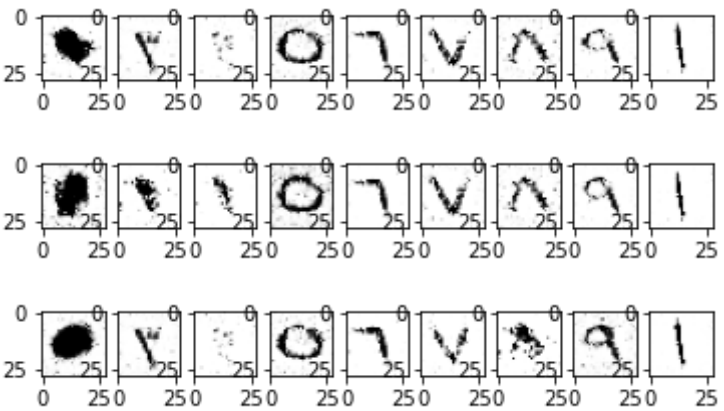
Using the level 1 labels obtained by the autoencoder based clustering, we train a CGAN over them.

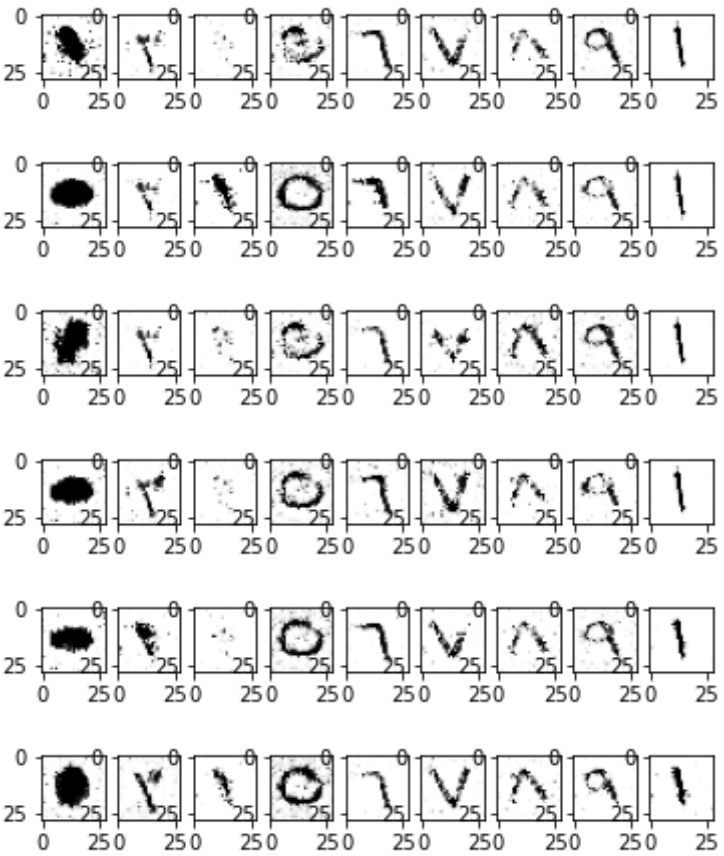
We for the most part sticked to the architecture proposed in paper but with a few modifications:

- We used soft labelling in the one-hot encoding of labels. We replaced 1 with 0.9 for the true label
- We use Leaky Relu instead of Relu
- We used the latent variable coming from a normal distribution not uniform
- We used RMSProp as optimiser
- We regularised over the penultimate features of discriminator for the real and generated images of same label

Learning Rate	G: 1e-4 D: 2e-4; Momentum 0.5 for both
Optimiser	RMS Prop
Regulariser	For a particular label, the penultimate features of the real and generated image should be similar. L2 loss is added over these features

Fake Images





FID:

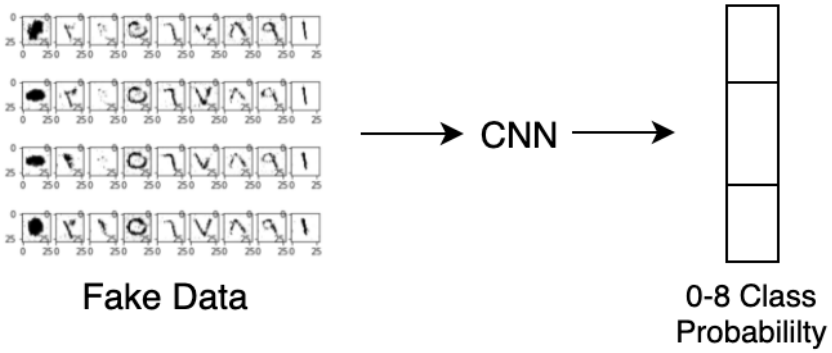
```
1 !python -m pytorch_fid --device cuda:0 /co

Downloading: "https://github.com/mseitzer/pyt
100% 91.2M/91.2M [00:00<00:00, 107MB/s]
100% 179/179 [00:44<00:00,  3.98it/s]
100% 180/180 [00:45<00:00,  3.97it/s]
FID:  189.60393837673294
```

For the trained generator, we generated 5000 images per class and then stored the generated images in a folder. The real images were stored in another folder. Then we train the pytorch_fid library to get the fid score. We got an FID score of 189.60 which is the best we got after trying a whole bunch of hyperparameter tuning.

Part 3
Recurrent Relational Network

Using the trained Conditional GAN, we generate a lot of data, and train a CNN classifier on it. This classifier is used to convert the query and target images into symbolic representation.

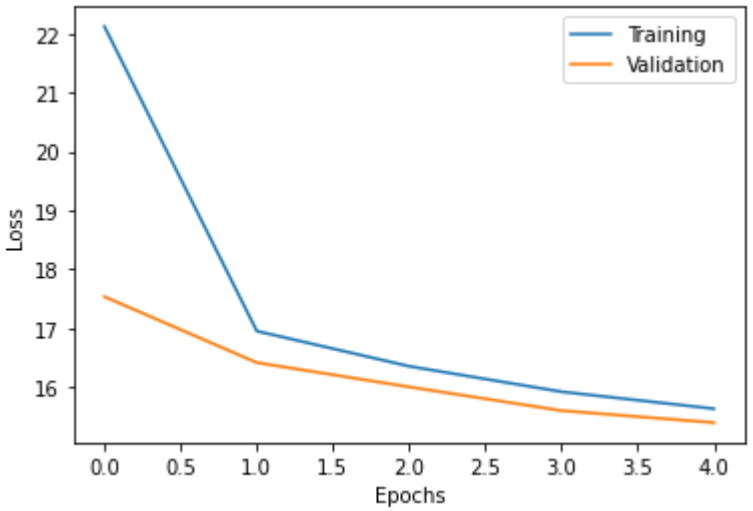


Now the data is in the form in which the RRN can process. We implement the RRN, and train it.

Hyperparameters

Optimizer	Adam
Learning Rate	0.0002
Batch Size	16
n_steps	30

Training



Performance

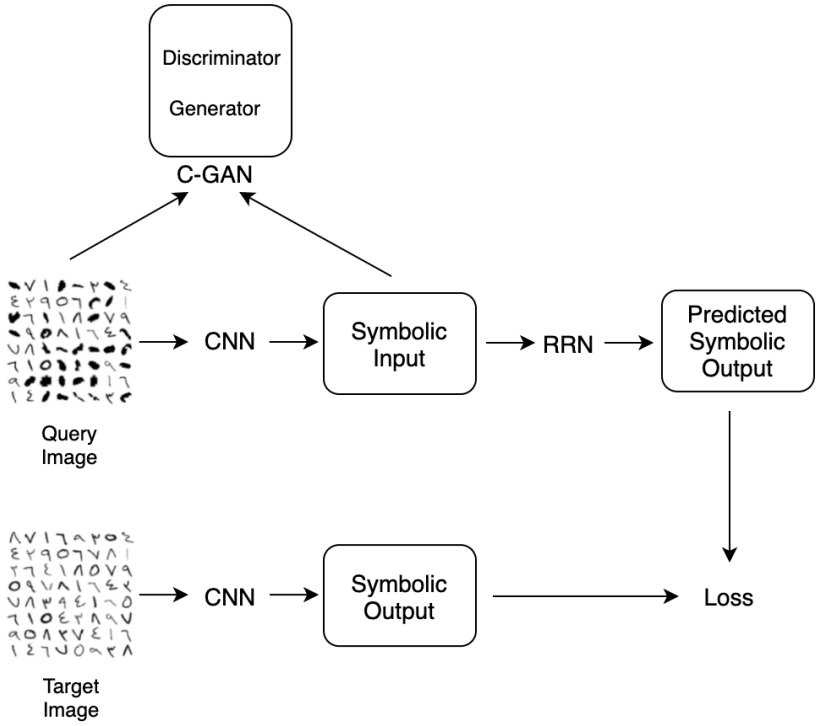
Due to noise in the input and output labels, the RRN could not train properly. The RRN could not classify even a single sudoku board correctly.

- Due to the high training time of RRN, we could not train it properly (10-12 minutes per epoch).

When checking element to element wise accuracy we found out that our model achieves 68% in the limited number of epochs we could train it. Approximately, 50% of the total were originally unknown in the query images.

Part 4
End to end (cGAN + RRN)

In this, we try to train an end-to-end model. The RRN, c-GAN and the CNN are trained all together. The diagram can be seen below :-



We combine 3 different models together :-

1) CNN
2) Conditional GAN
3) Recurrent Relational Network

CNN

Takes a query / target image as an input and outputs a symbolic representation of the image.

Optimizer	Adam
Learning Rate	0.01
Weight Decay	0.0001

RRN

Takes, a matrix of sudoku as input, and outputs a solved sudoku matrix

Optimizer	Adam
Learning Rate	0.00002
Weight Decay	0.0001

C-GAN

Takes the image and labels output by CNN, as input and learns to generate conditional data

- **Generator**

Optimizer	RMS prop
Learning Rate	0.00001
Weight Decay	0.0001

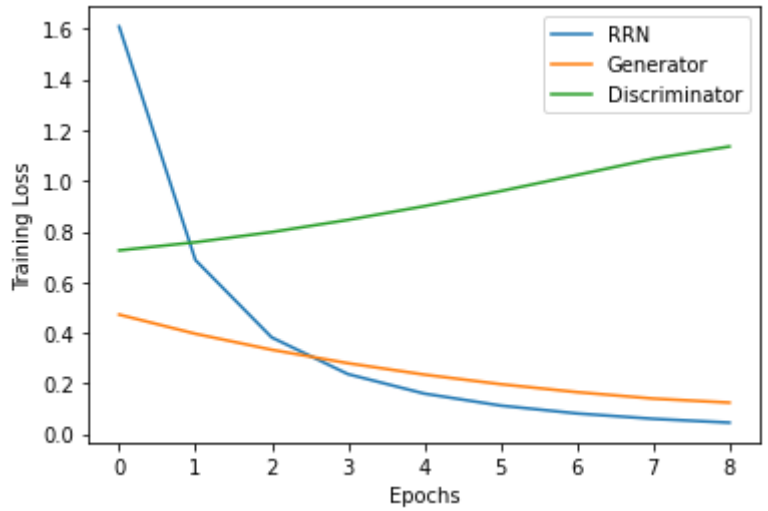
- **Discriminator**

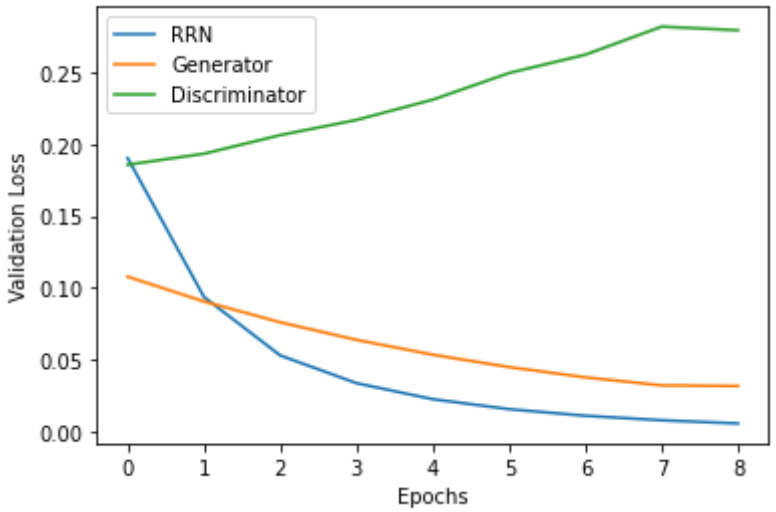
Optimizer	RMS prop
Learning Rate	0.00002
Weight Decay	0.0001

Loss

Loss = (loss due to gan + classification loss at the output of RRN)

Training





We have implemented the above model and tried to train it. The loss decreased over epochs, however, we could not ascertain any theoretical guarantees of the loss function reaching to the required equilibrium. Due to high training time per epoch (10-12) we could not train it properly.

- Due to the high training time of the combined model, we could not train it properly (10-12 minutes per epoch).

References

[1] autoencoder (toronto.edu) - Our architecture of autoencoder was motivated from their implementation