SMAI PROJECT on Deep learning

Common Representation Learning(CRL)

presented by Team - ThreeChums

- Anchal Soni (2020201099)
- MK Utkarsh (2020201027)
- Varun Nambigari (2020201079)

Common Representation Learning(CRL)

Using Step-based Correlation Multi-Modal CNN

PPT WALKTHROUGH

- 1) Introduction
- 2) Implementation of the paper
- 3) Test and Result
- 4) Individual contribution

Common Representation Learning(CRL)

- learning a common representation for multi-view data, wherein the different modalities are projected onto a common subspace
- For example, task of abstract scene recognition in a movie

In this paper we construct a novel step-based correlation multi-modal CNN (CorrMCNN) which can reconstruct one view of the data given the other.

Dataset Used: MNIST Handwritten Digits

MNIST contains

- Total 70,000 images
- Training data = 60,000
- Testing data = 10,000
- The images are grayscale, each 28x28 pixels
- Each image is sliced into two for two input views, each 28x14 pixels

Architecture of CorrMNCC

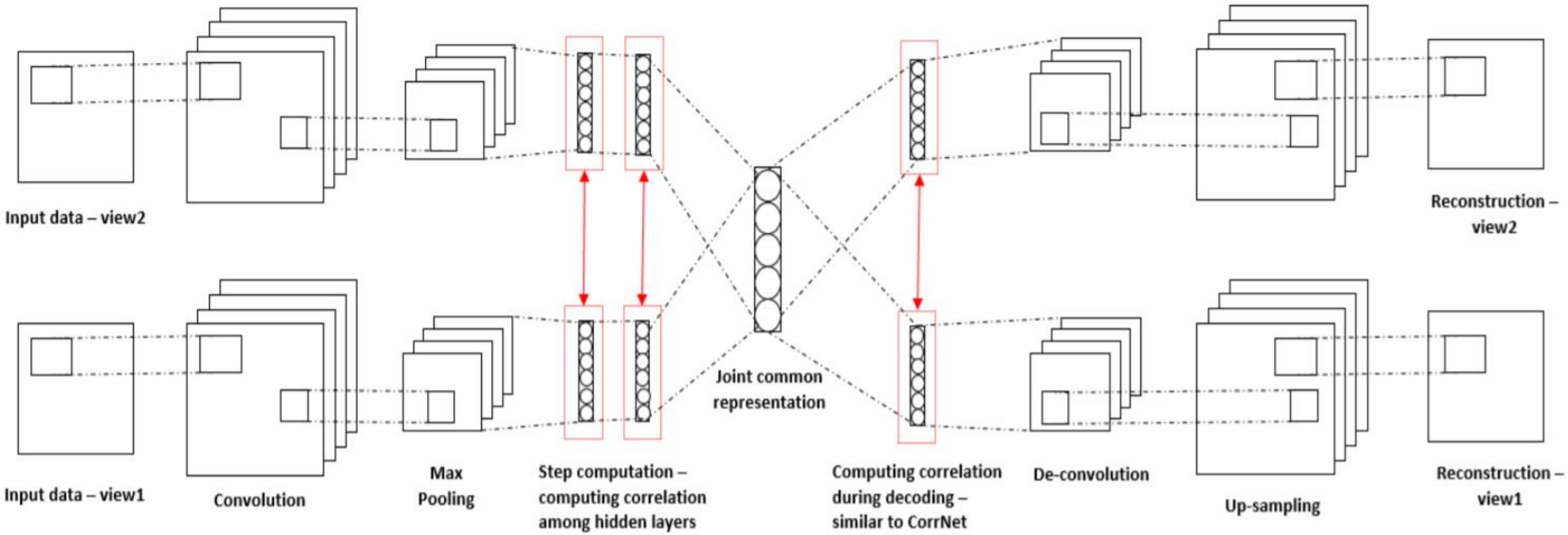


Fig. 1. Overview of the CorrMCNN. The bidirectional arrows shows the step correlation computation and cross-reconstructions at the intermediate steps.

$$L_1 = \sum_{i=0}^{N} L(z_i, g(h(z_i))$$

$$L_2 = \sum_{i=0}^{N} L(z_i, g(h(x_i))$$

$$L_3 = \sum_{i=0}^{N} L(z_i, g(h(y_i))$$

$$L_4 = \sum_{k=0}^{K} \sum_{i=0}^{N} L(h(x_i^k), h(y_i^k))$$

$$L_5 = \sum_{i=0}^{N} L(g(h(x_i)), g(h(y_i))$$

Self,cross reconstruction Losses and step computation:

- zi = {xi; yi}, where zi is the concatenated representation of input views xi and yi
- L is the mean square error function
- g, h are non-linearities generally taken as sigmoid or reLU
- g(h(xki)) and g(h(yki)) are the hidden representations
- K represents the kth intermediate hidden layer

Correlation loss:

$$L_6 = \lambda \; corr(h(X), h(Y))$$

$$L_7 = \sum_{k=0}^{K} \lambda_k \; corr(h(X^k), h(Y^k))$$

$$\mathrm{corr}(h(X), h(Y)) = \frac{\sum_{i=1}^{N} (h(\mathbf{x}_i) - \overline{h(X)})(h(\mathbf{y}_i) - \overline{h(Y)})}{\sqrt{\sum_{i=1}^{N} (h(\mathbf{x}_i) - \overline{h(X)})^2 \sum_{i=1}^{N} (h(\mathbf{y}_i) - \overline{h(Y)})^2}}$$

$$L(\theta) = \sum_{i=0}^{5} L_i - \sum_{j=6}^{7} L_j$$

Finally, the CorrMCNN is optimized using this function

- where θ are the parameters of CorrMCNN
- Here, we minimize the self-reconstruction and cross-reconstruction
- and maximize the correlation between the views.

Model Summary

- Two input channels
- In each channel:
  * Two convolution layers
  * with MaxPooling layer
  * and Batch Norm layer
  * two fully connected layer
- Joint common representation with 50 dimension
- For each projection:
  * Upsampling
  * Deconvolution layer
- Two final reconstructed views

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 128, 26, 12]	1,280
MaxPool2d-2	[-1, 128, 13, 6]	0
BatchNorm2d-3	[-1, 128, 13, 6]	256
Conv2d-4	[-1, 64, 11, 4]	73,792
MaxPool2d-5	[-1, 64, 5, 2]	0
BatchNorm2d-6	[-1, 64, 5, 2]	128
Linear-7	[-1, 500]	320,500
Dropout-8	[-1, 500]	0
Linear-9	[-1, 300]	150,300
Conv2d-10	[-1, 128, 26, 12]	1,280
MaxPool2d-11	[-1, 128, 13, 6]	0
BatchNorm2d-12	[-1, 128, 13, 6]	256
Conv2d-13	[-1, 64, 11, 4]	73,792
MaxPool2d-14	[-1, 64, 5, 2]	0
BatchNorm2d-15	[-1, 64, 5, 2]	128
Linear-16	[-1, 500]	320,500
Dropout-17	[-1, 500]	0
Linear-18	[-1, 300]	150,300
Linear-19	[-1, 50]	15,050
Linear-20	[-1, 294]	14,994
Upsample-21	[-1, 3, 26, 12]	0
ConvTranspose2d-22	[-1, 1, 28, 14]	28
Linear-23	[-1, 294]	14,994
Upsample-24	[-1, 3, 26, 12]	0
ConvTranspose2d-25	[-1, 1, 28, 14]	28

Parameters used

Value of lambda for correlation loss:

- lambda 1 = 0.02
- lambda2 = 0.003
- lambda3 = 0.05

No. of epochs

- 50

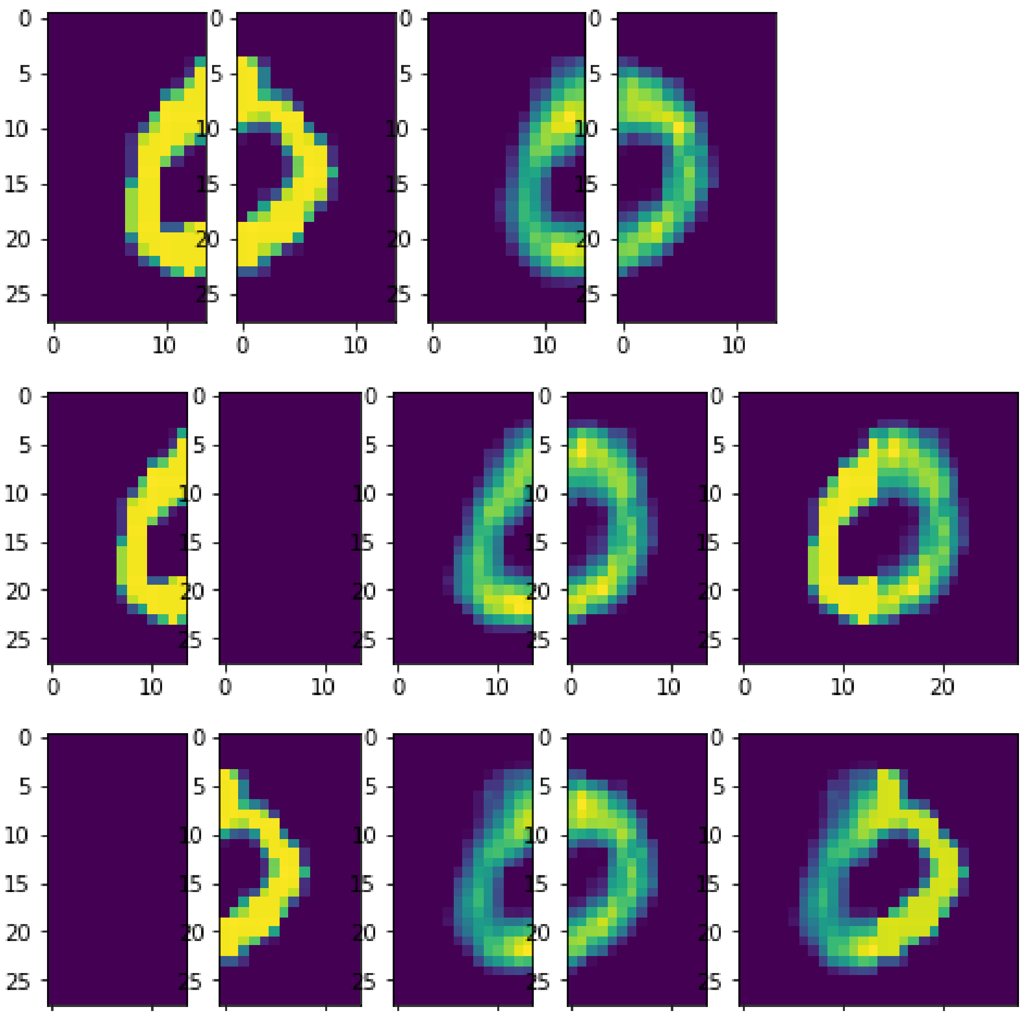
Optimizer used

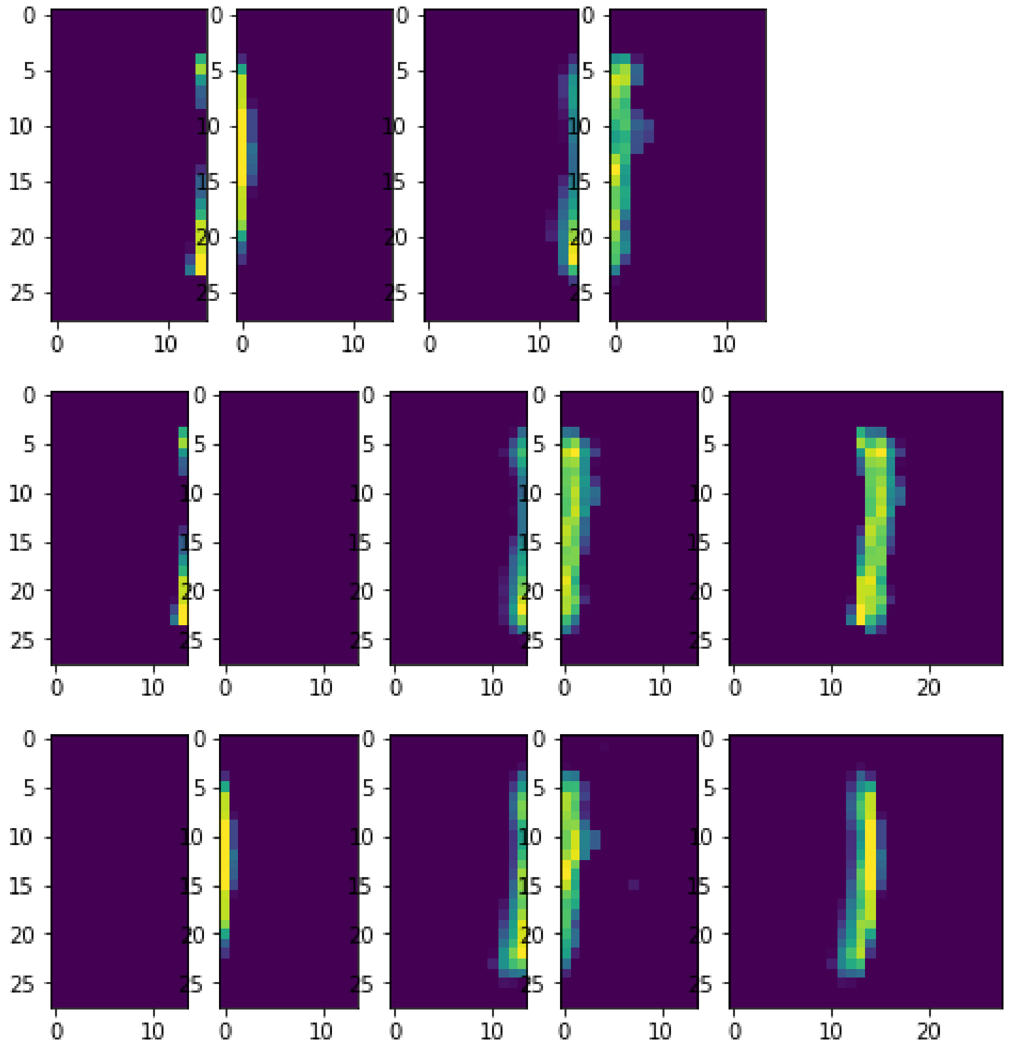
- Adam
- Learning rate = 0.01

Activation function used

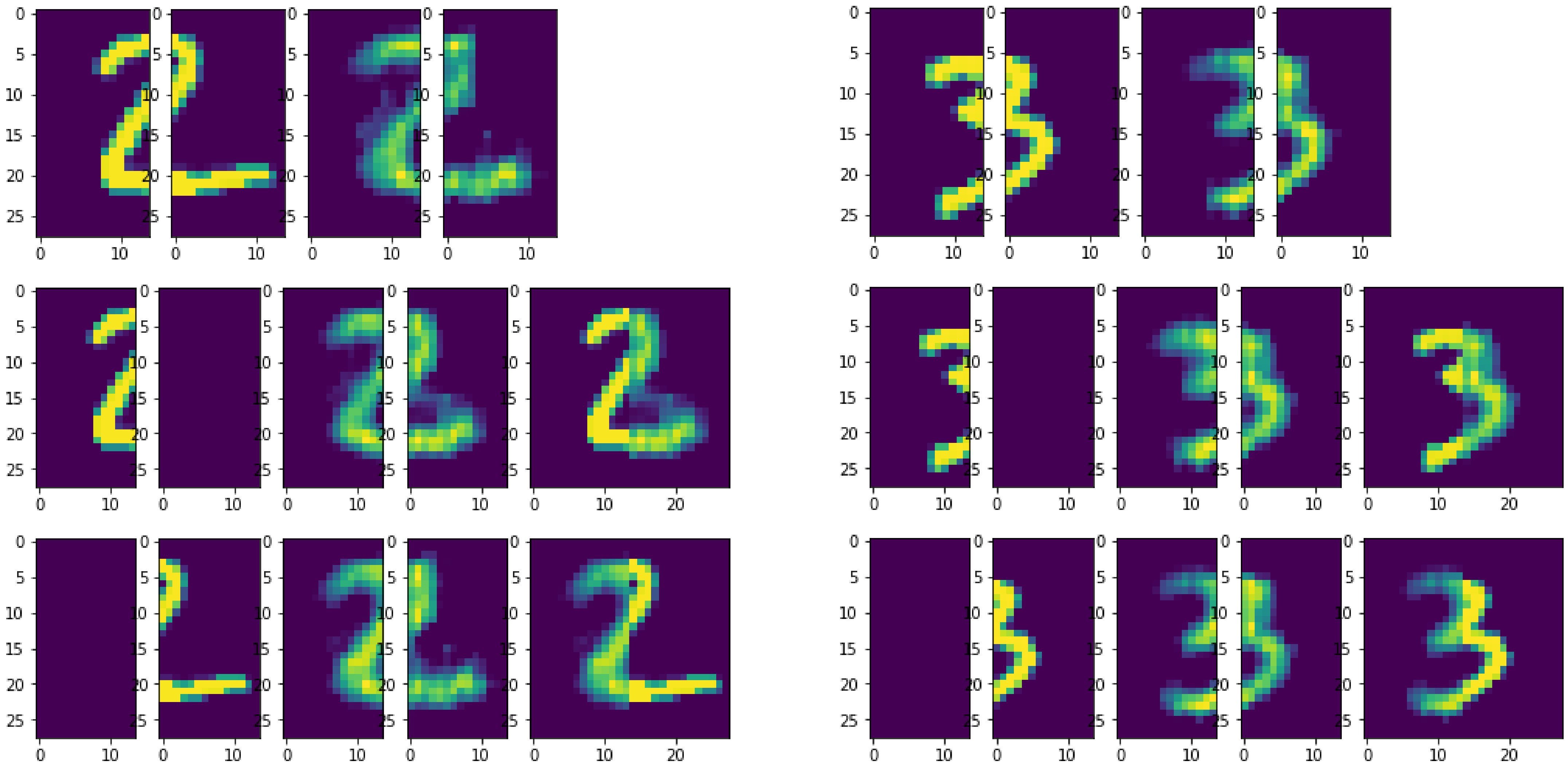
- reLU

Reconstruction of '0' and '1'

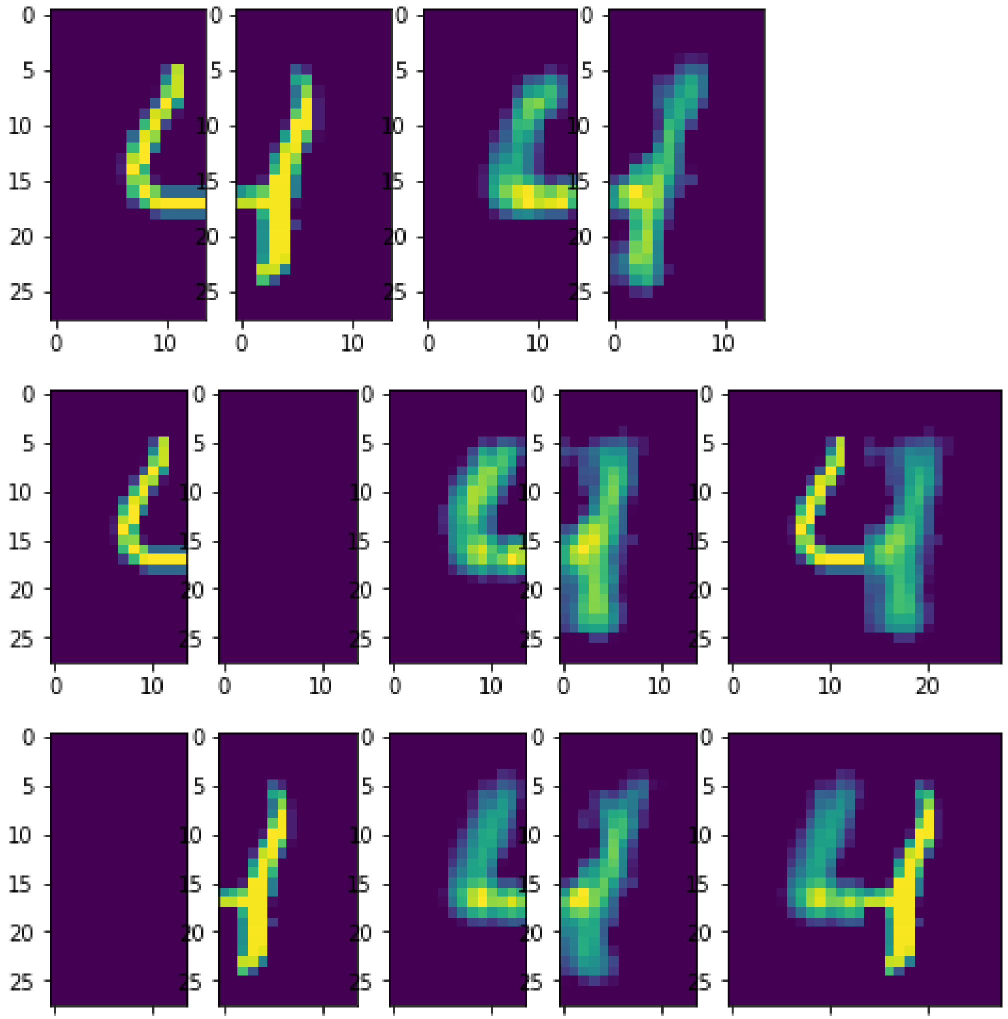


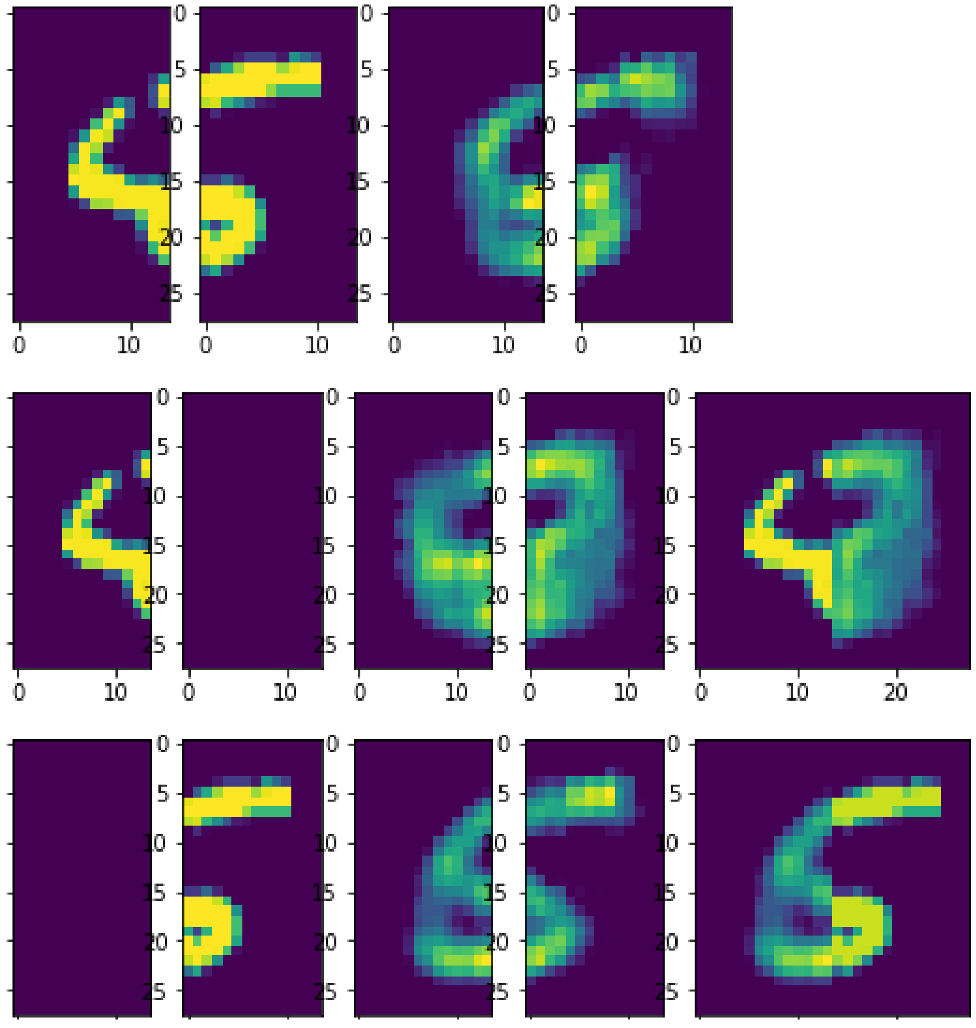


Reconstruction of '2' and '3'

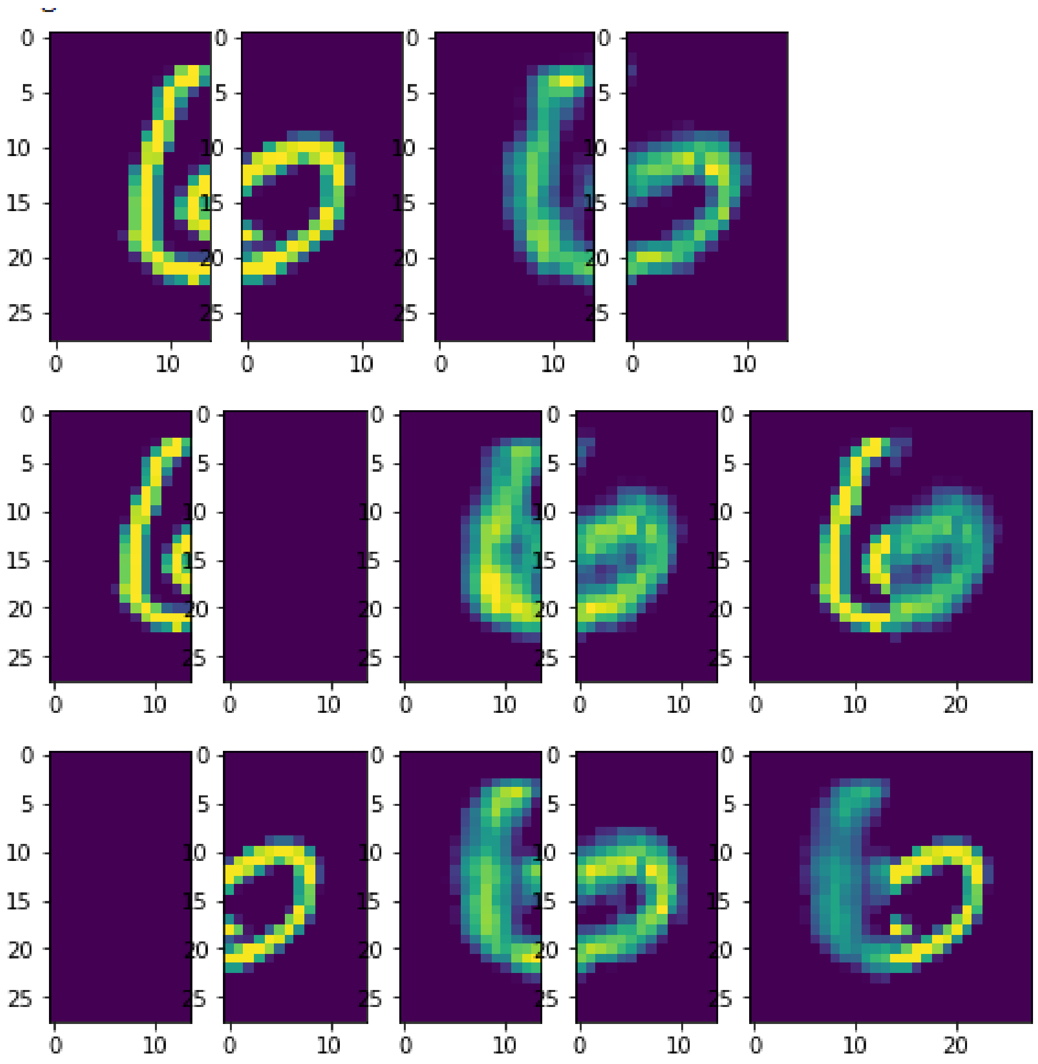


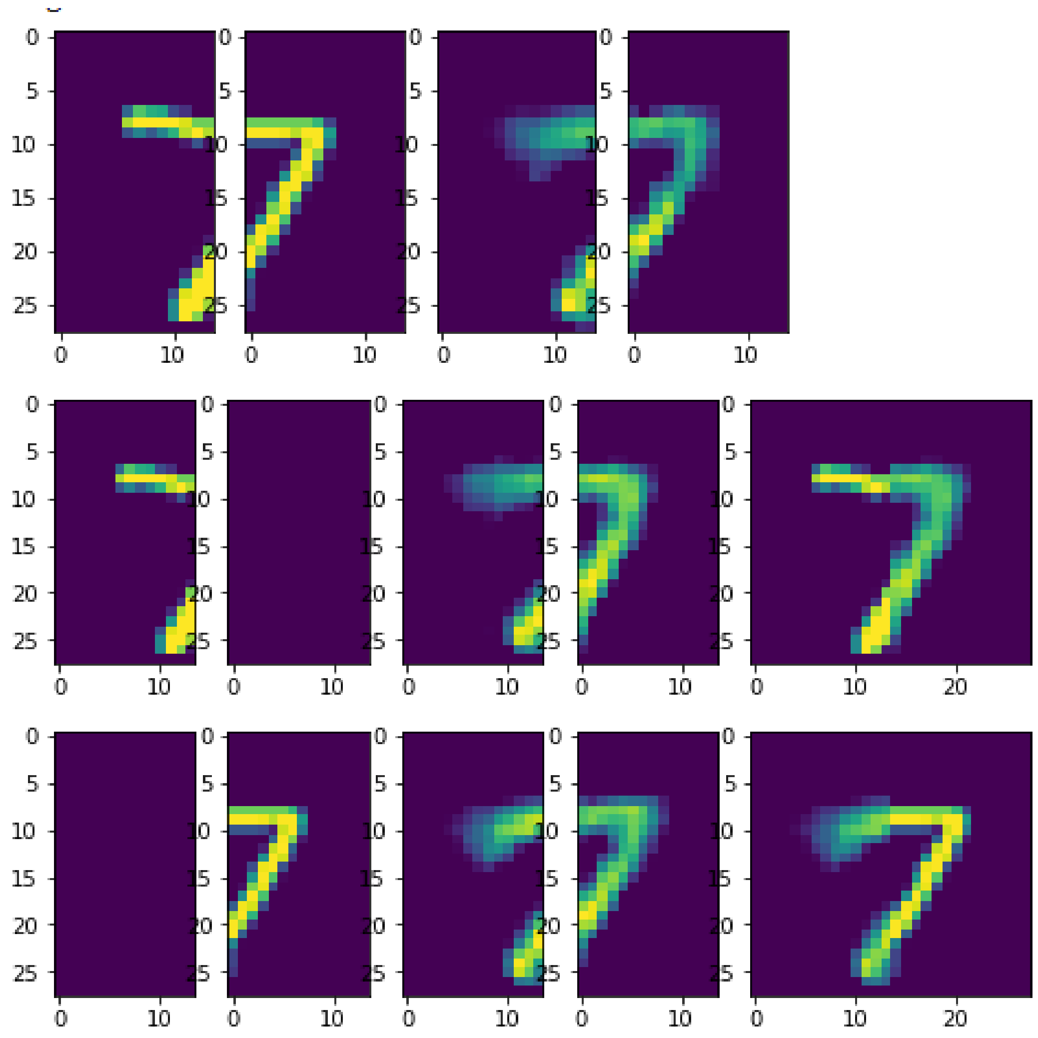
Reconstruction of '4' and '5'



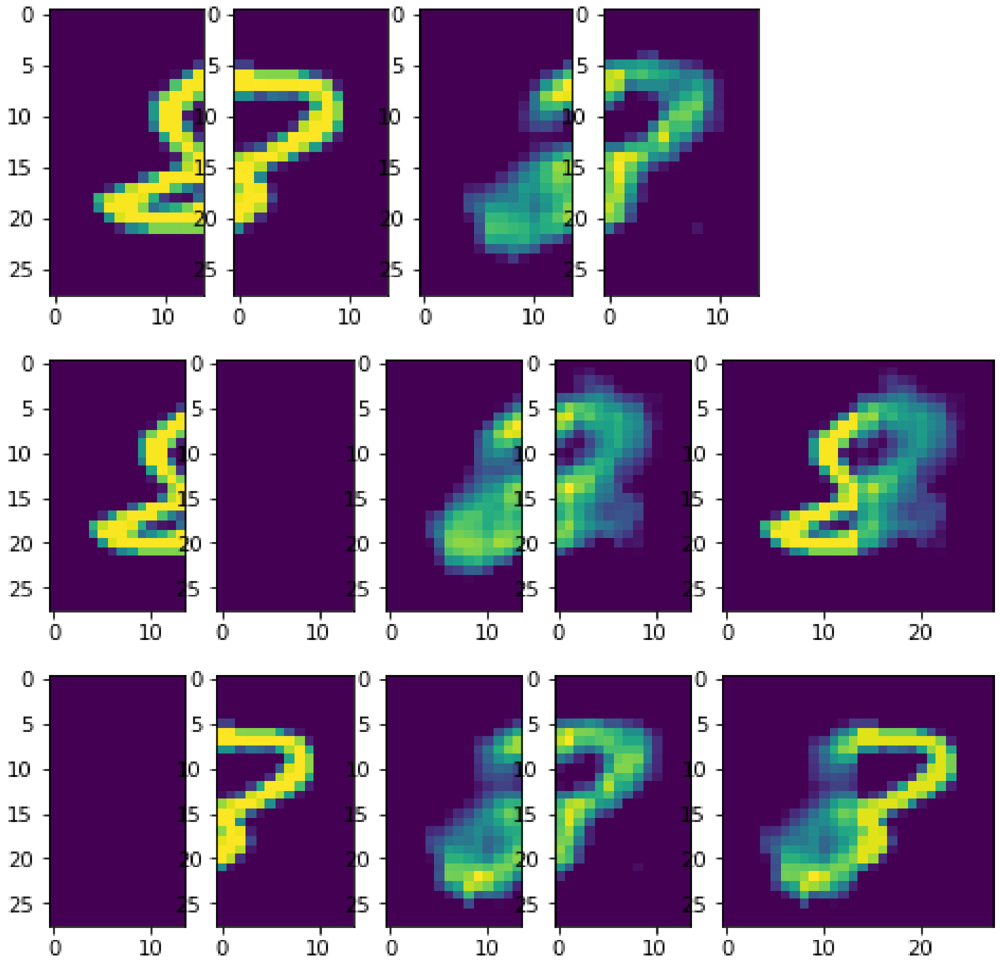


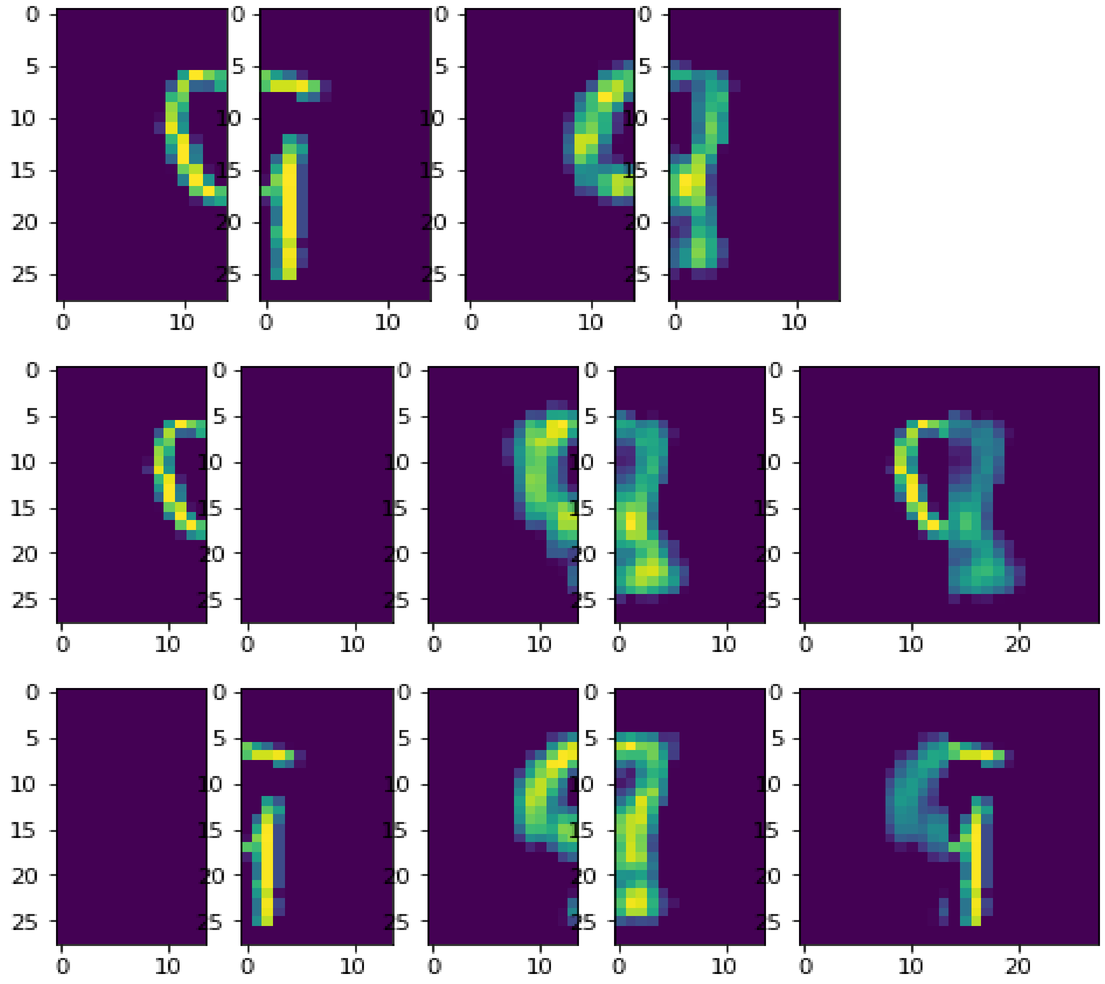
Reconstruction of '6' and '7'





Reconstruction of '8' and '9'





Testing the model

- Trained a (2 layer fully connected MLP) classifier with input as hidden representation of the constructed using both the views.
- For testing constructed the hidden layer from only one view. Then classified based on the constructed hidden layer.
- Results:

  Classification accuracy from only left view: 74.25%
  Classification accuracy from only right view: 77.68%

Individual Contribution:

- Anchal Soni (2020201099) - Data preprocessing. Built an Autoencoder (First three losses)
- Utkarsh MK (2020201027) - Corr loss function, loss 4 to loss 7. Completed the basic model class.
- Varun Nambigari (2020201079) - Trained the model, plotted the results. Tested using a classifier.

THANK YOU

Jupyter notebook link:
https://colab.research.google.com/drive/1hUFOlHfxVyqbVWP0Spciley-K-sJiLI-?usp=sharing

Research paper link:
https://arxiv.org/pdf/1711.00003.pdf