CSCI 545 **Big Data Analytics**
Course Project Report

Meruyert Mussakhanova, Yerlan Amanzholov, Zhaniya Koishybayeva

Task description

We were given a set of 32x32 images (9200 in total) to train an algorithm to complete the images from a testing set that had missing 16x16 pieces in the center.

Methodology and results

In order to explore the dataset, our first method was rather easy, we tried to substitute the missing square with the one corresponding to the image with the least MSE between the "framings" of images. The problem with this algorithm was its inconsistency of successive substitutions. If the closest image pictured the same symbol, the region would fit almost perfectly to complement it (Fig. 1)

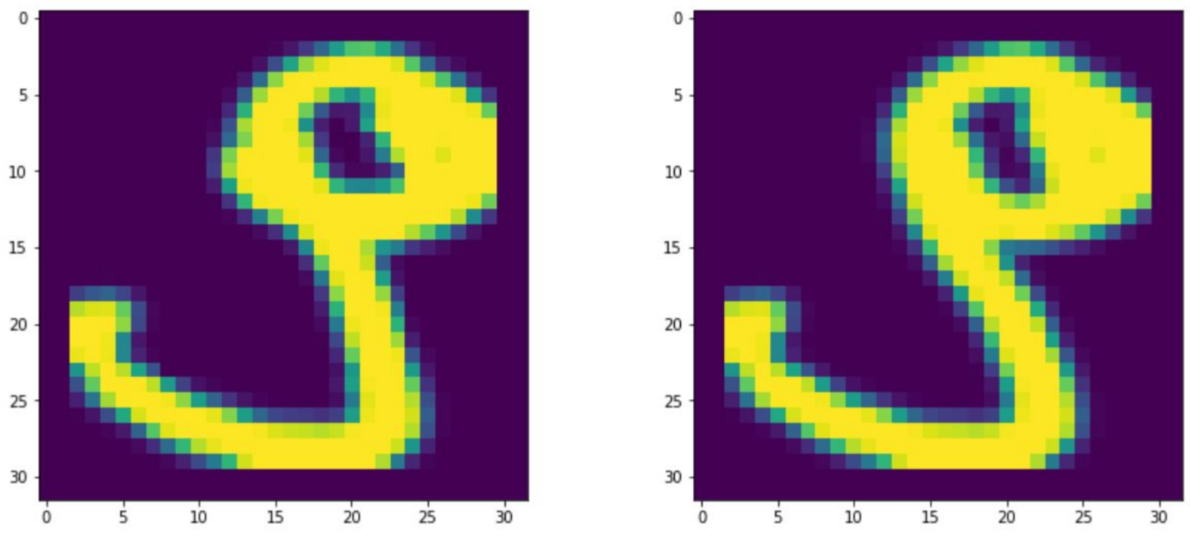

Figure 1. Fitted image on the left and true image on the right.

However, if the closest symbol wouldn't match the one under consideration, the algorithm returned very messy outcome (Fig. 2)

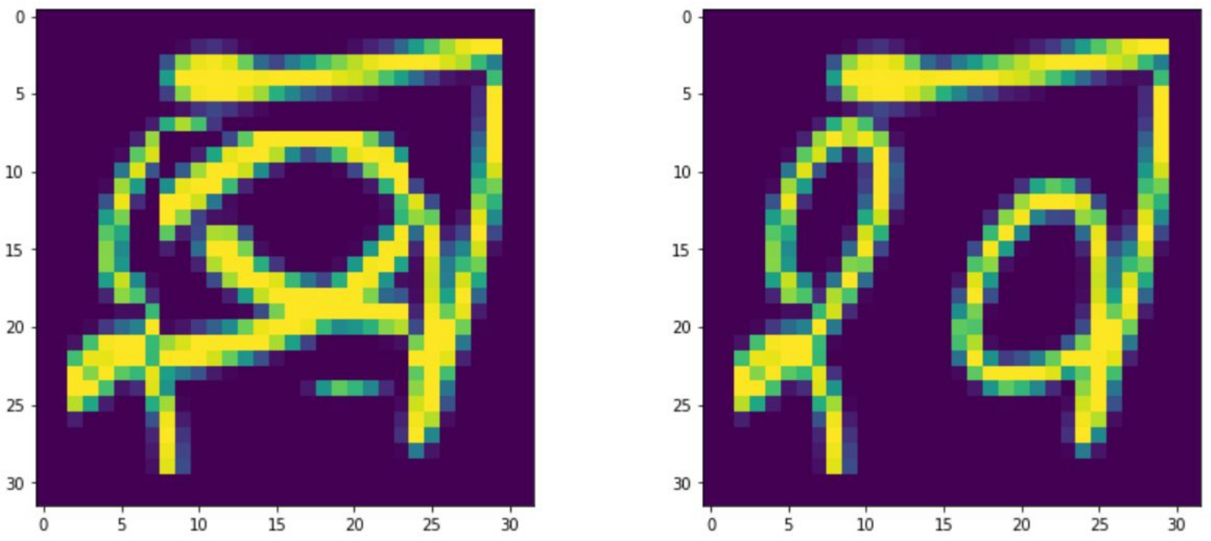
Figure 2. Fitted image on the left and true image on the right.

This method resulted in an average MSE of 0.1909.

Therefore we decided to generalize the previous method, implementing the kNN algorithm. We figured out that the total number of classes in the training set was 46 symbols, 200 of each, so after assigning the classes, we tested the kNN algorithm for k=3, k=5 and k=7. Unfortunately, even less predicted images would make sense this time (Fig. 3), the average MSE was increasing and exceeded 0.3. Therefore, we moved to more complex solutions.

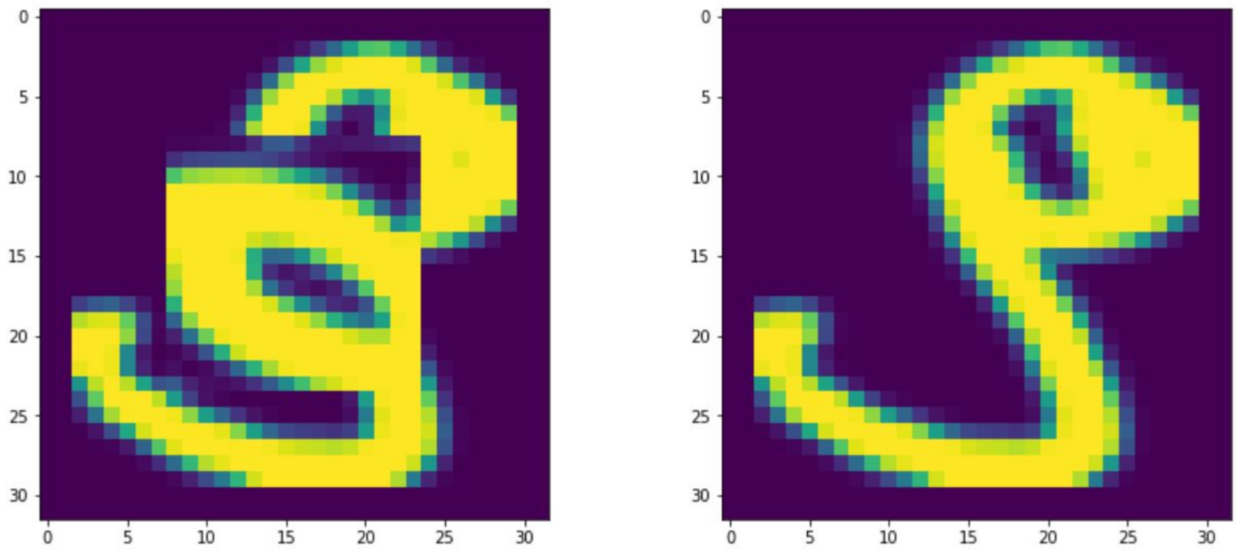
Figure 3. Fitted image on the left and true image on the right.

The group chose to try out simple neural approaches to this problem. The first choice was to construct a convolutional neural network since the task is connected with images. The model consists of convolutional layers with different kernel sizes, which squeezed the 32x32 image to 16x16. The network was trained end-to-end using MSE loss. The schematic can be seen on Fig. 4.

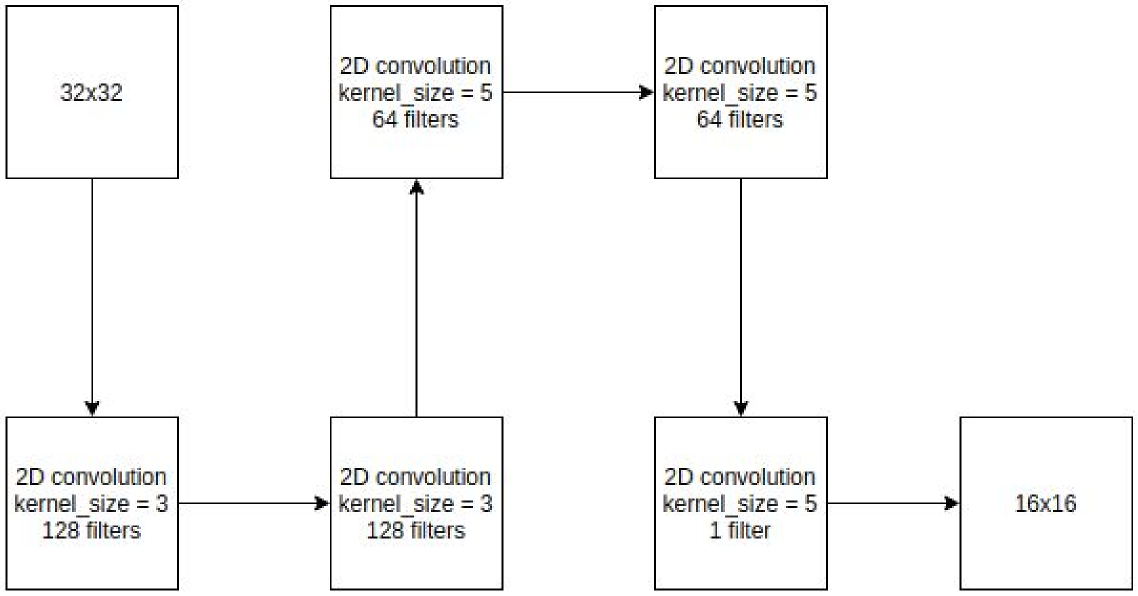

Figure 4. Architecture of CNN network

Validation loss of this network on 5-fold cross validation was 0.1281. However, due to locality of the filters in CNN, the network was able to reconstruct only boundary pixels of the target region. The example can be seen on Fig. 5.

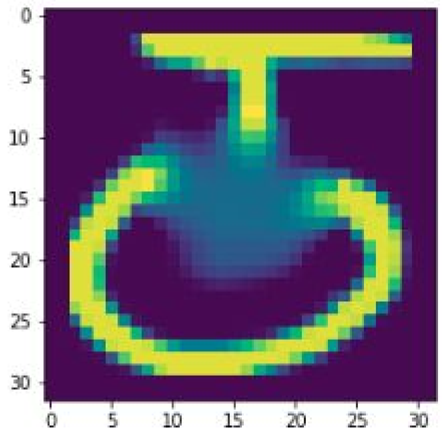

Figure 5. Example reconstruction using CNN.

The next approach was to construct a fully connected neural network. One improvement of this model is that it takes only frame pixel values, omitting the middle part, which was replaced by ones. Chosen architecture can be seen on Fig. 6 below.

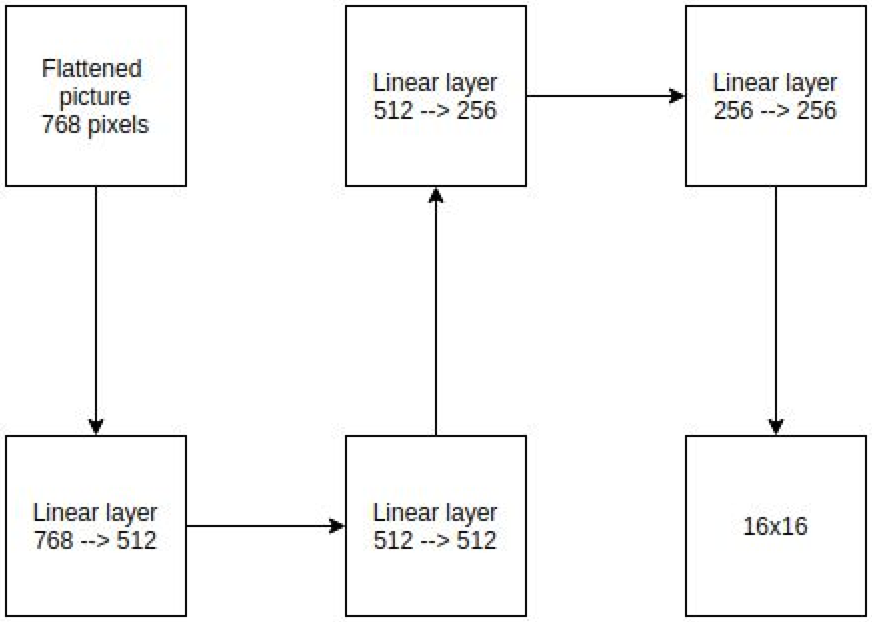

Figure 6. Fully connected neural network architecture

Validation loss on 5-fold cross-validation was 0.1514. Despite the loss, which is higher than CNN, fully-connected NN has shown good results during generation, being able to capture general patterns. The example can be seen on Fig. 7.

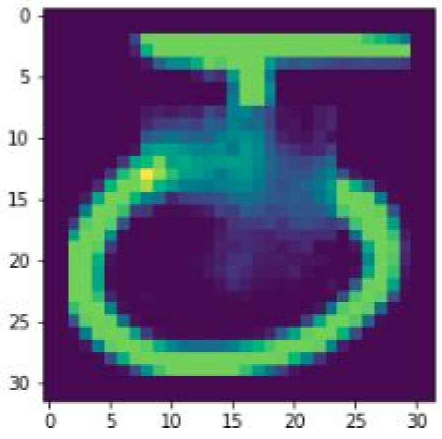

Figure 7. Example reconstruction using NN.

Project structure

Project

```
├── CNN
│   ├── config.py - Configuration file with parameters
│   ├── dataset.py - Dataset file for PyTorch to preprocess data
│   ├── engine.py - Engine with train and evaluation functions
│   ├── inference.py - File to generate prediction on test set
│   ├── model.py - File with model architecture
│   ├── models - Folder with trained models
│   ├── train.py - File to train the model
│   ├── Untitled.ipynb - Some visualizations to track results
│   └── utils.py - Utility functions
├── data
│   ├── bda-image-challenge-row-testdist.txt
│   └── bda-image-challenge-train.txt
├── DNN
│   ├── config.py
│   ├── dataset.py
│   ├── engine.py
│   ├── inference.py
│   ├── __init__.py
│   ├── model.py
│   ├── models
│   ├── train.py
│   ├── Untitled.ipynb
│   └── utils.py
├── predictions.csv - Final predictions on test set generated by our team
├── README.rst - How to re-run the scripts and train models
└── requirements.txt - Required python packages
```

Way forward

The implemented simple CNN-based model allowed to achieve feasible results, however the group has researched the area and found some more advanced solutions that could provide better results but needed more time to implement. Pathak et al. [1] developed Context encoder with L2 loss + Adversarial loss that allowed to fill the hole in an RGB image with a picture that is visually acceptable. The encoder-decoder pipeline consists of regular CNN layers. The researchers also compared L2 loss and combined L2 + Adversarial loss and demonstrated that the latter achieved much sharper results.

References

[1] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell and A. A. Efros, "Context Encoders: Feature Learning by Inpainting," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 2536-2544.