**Project Proposal**

**Title:**

Catch These Signs

**Team Members:**

Devon Tyson, Philip Cesani, and Randall Harrison

**GitHub:**

https://github.com/Catch-These-Signs/Sign-Language-Project

**Introduction:**

Sign languages are languages that use the visual-manual modality to convey meaning, expressed through manual articulations in combination with non-manual elements. They are full-fledged natural languages with their own grammar and lexicon. American Sign Language is one of the most commonly used minority languages in the United States. Therefore, there is a demand for technology that can assist the deaf community and those who use sign language with a broader means of communication.

**Problem Description:**

The goal of the project is to better understand and improve communication for the deaf community by using machine learning and computer vision techniques. We will achieve this by using images of signed letters and translating them into text characters that are readable by a computer. We will use a Convolutional Neural Network to take images of signed letters and make predictions about them in order to classify them as their respective letter.

**Description of Data:**

The American Sign Language letter database of hand gestures represent a multi-class problem with 24 classes of letters (excluding J and Z which require motion). Our dataset format is patterned to match closely with the classic MNIST dataset. Each training and test case represent a label (0-25) as a one-to-one map for each alphabetic letter A-Z.

**Our process:**

We reviewed many projects, chose the sign language dataset and analyzed the data within the set. We then determined the attributes and classifications for the project. The attributes include the pixel value for each pixel in each image and the classifications include the corresponding letter of the alphabet. We have collected and prepared the data for use and set up a GitHub organization with a repository for the project. We have downloaded the proper software, reviewed the Python programming language and have begun working on the project in Jupyter Notebook. For the next step, we built the Convolutional Neural Network, Multi-Layer Perceptron, and K-Nearest Neighbor algorithms. We set up all of the training parameters, found the placeholders, biases, and weights to use to train the data for each neural network. Each algorithm was set up with its own loss function and optimizer. We set up the Convolutional and Pooling layers for the CNN algorithm and began training. Once the data was trained for each model, we tested the data and made predictions.

**Preliminary Plan:**

1. Find a project
2. Understand the problem
3. Download the proper software
4. Review and learn the Python language
5. Set up a GitHub Organization with repository
6. Define the project objective
7. Prepare the data
8. Collect the data
9. Select Algorithms
10. Train the model
11. Test the model
12. Make predictions
13. Conclude the results

**Methods Overview:**

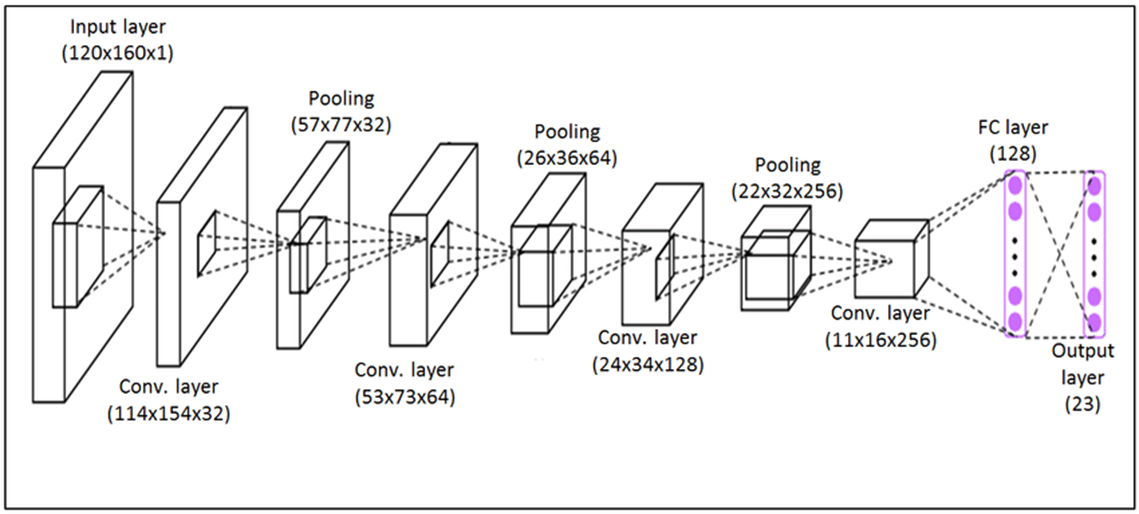
We used the sign language data from MNIST with three different algorithms, Convolutional Neural Network, Multilayer Perceptron, and k-Nearest Neighbors in order to compare the results from them.

**Convolutional Neural Network (CNN):**

CNN is a good choice for image recognition because the weights of the convolutional layer being used for feature extraction as well as the fully connected layer being used for classification are determined during the training process. In traditional models for pattern recognition, feature extractors are hand designed.

CNN compares images piece by piece. The pieces that it looks for are called features. Rough feature matches are found in roughly the same position. It then creates a map to put the values of the filter. Sliding the filter throughout the image to get an output of values in a matrix. ReLU layer (Rectified Linear Unit) a transform function that removes negative values from the matrix is then implemented. A Pooling layer is used which shrinks the image into a smaller size. Stacking the layers is done by adding additional layers of Convolution, ReLU, and Pooling which will reduce the output matrix further. The Fully Connected layer is the final layer where the actual classification happens. Here we take our filtered and shrunken images and put them in a single list or vector. When we feed an image there will be some element in the vector that will be high that will identify a match with some accuracy.

The Design Architecture of CNN is as follows:



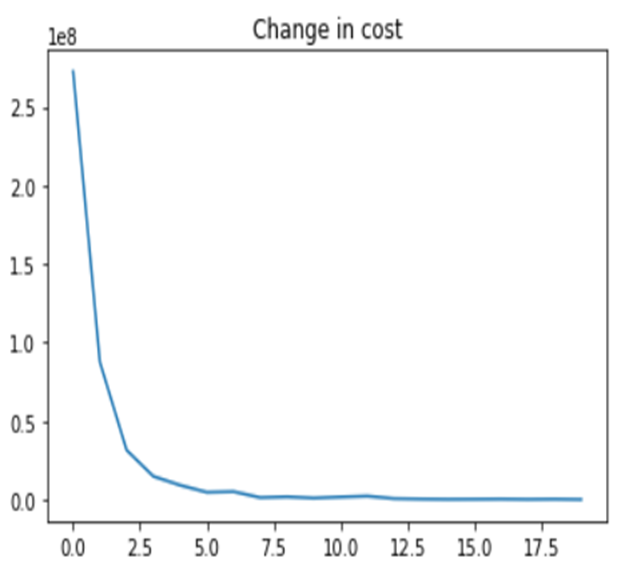
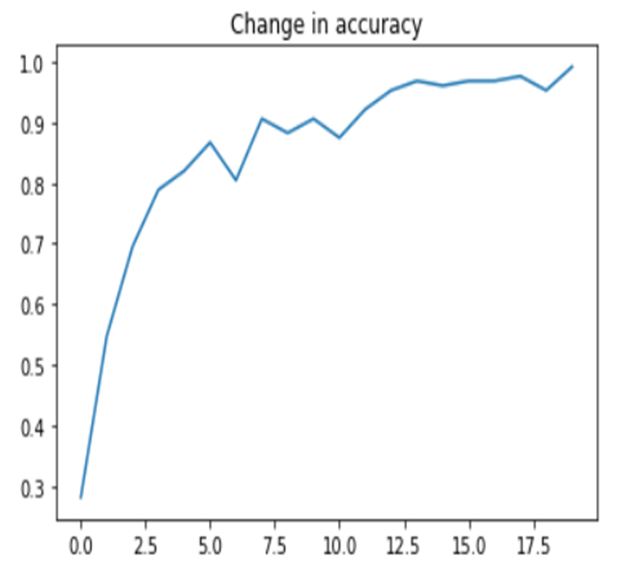
In the CNN Neural Network, we used the following:

Loss function: Catagorical Cross Entropy  
Optimizer: Adam  
Activation Function: Relu (Rectified linear unit)  
Layers Used: 10  
Learning Rate: 0.001  
Epochs: 2000

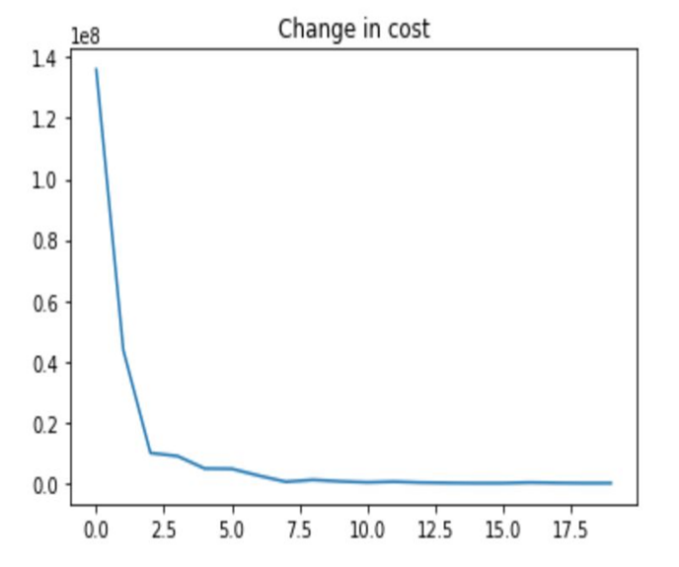
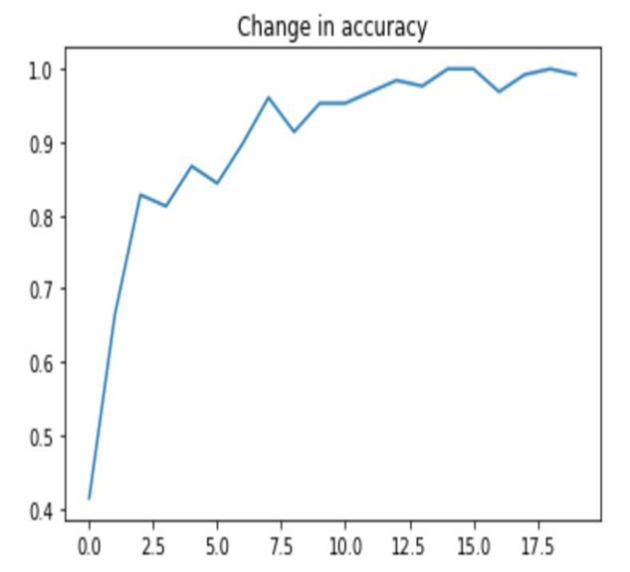
Results for splitting dataset into training and holdout(test set) – ratio [50%/50%]

Accuracy on Training Data: 97.83%

Accuracy on Test Data: 96.92%



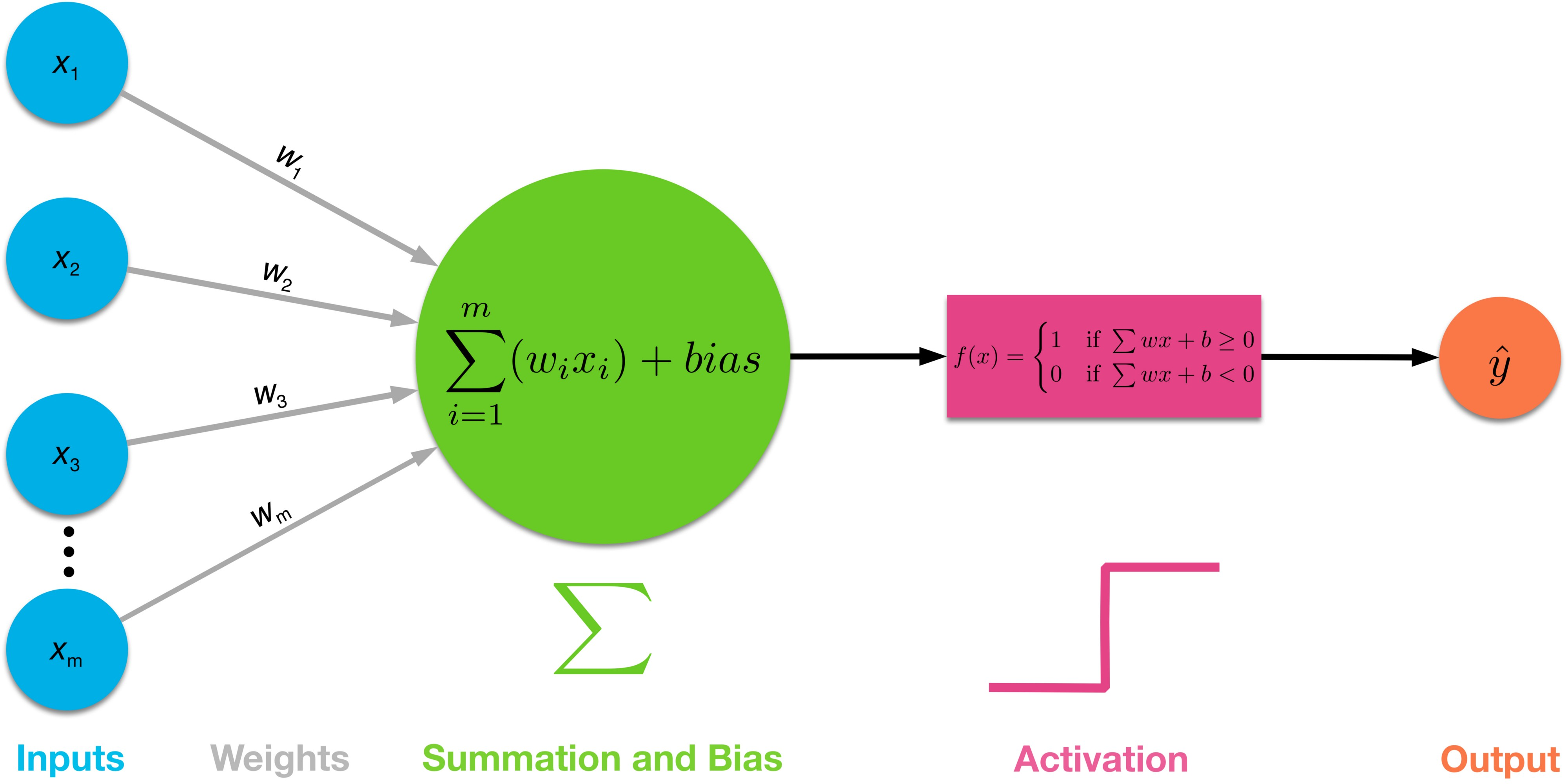
Accuracy after training on whole dataset: 99.05%



**Multi-Layer Perceptron (MLP):**

Multi-Layer Perceptron is the predecessor to Convolutional Neural Networks. It was once used as image classification due to the use of hidden layers in the neural network. It's good for image classification because it can distinguish data that is not linearly separable. MLP works as a feedforward artificial neural network. It contains many perceptrons, or binary classifiers, that are organized into layers. There are at least 3 layers included – the input layer, a hidden layer and an output layer; however, there may be many hidden layers. Each node is a neuron that uses an activation function in each layer. In our case, we used ReLu activation function. Backpropagation is used for training, which uses the previous layer as an input for the current layer. Learning occurs in the perceptron by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. Once all layers have been activated, linear algebra reduces the layers into a two-layer input-output model.

Each layer can be visualized like the example below:



In the MLP neural network, we used the following:

Loss function: Cross Entropy

Optimizer: Adam

Activation Function: ReLu

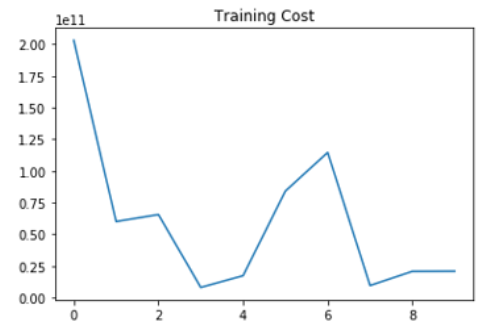
Layers: 10

Learning Rate: 0.002

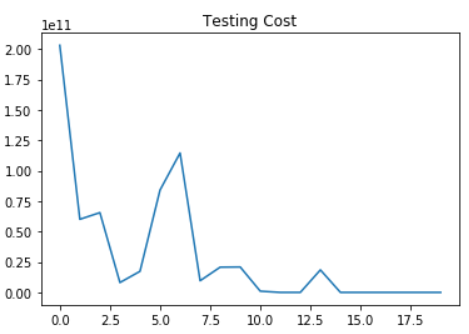
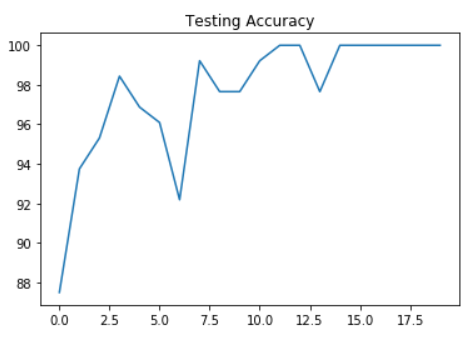
Batch size: 128

Epochs: 5000

The average training accuracy came out to 96.35%



The average testing accuracy came out to 99.14%



**K Nearest Neighbors (KNN):**

// description

**Conclusion:**

Multi-Layer Perceptron has little to no accuracy when it comes to complex images having pixel dependencies throughout. Convolutional Neural Networks are able to successfully capture the spatial and temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better. The role of CNN is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. K-Nearest Neighbor and Convolutional Neural Networks perform competitively, while CNN produces higher accuracy than KNN. For this dataset, a Convolutional Neural Network performs best.

**Future work:**

A visual recognition algorithm for American Sign Language could provide new benchmarks that challenge modern machine learning methods such as Convolutional Neural Networks and could also help the deaf and hard-of-hearing to better communicate by using computer vision applications. In the future, the results from this project can be used in collaboration with a text-to-speech application with the intention of making communication as easy as possible for the deaf community. It could potentially even make communication faster and easier during conversations between deaf and blind people together.

**References:**

1.Huy V. Vo, Francis Bach, Minsu Cho, Kai Han, Yann LeCun, Patrick Perez, Jean Ponce.  Unsupervised Image Matching and Object Discovery as Optimization. In CVPR, 2019.

2. P. Isola, J-Y. Zhou, and A.A. Efros. Image-to-image translation with conditional adversarial networks. In CVPR, pages 1125 - 1134, 2017.

3. J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image to image translation using cycle-consistent adversarial networks. In CVPR, pages 2223–2232, 2017.

4. M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-toimage translation networks. In Advances in Neural Information Processing Systems, pages 700–708, 2017

5. Raia Hadsell, Sumit Chopra, Yann LeCun.  Dimensionality Reduction by Learning an Invariant Mapping.  In CVPR, 2006.