Final Project Report
Sign Language Classification

Chinmay Bhagwat, Suyog Khadke, Spyridon Kaperonis
CPE-646 Pattern Recognition and Classification, Dr. Hong Man

***Abstract*—In the United States alone there are 11.500.000 people with some sort of hearing impairment. Sign Language detection can improve the communication between deaf and the general population. We are pleased to present the implementation of sign language recognition using three different models. Convolutional Neural Network, Support Vector Machine, and K-Nearest Neighbors. Using these algorithms we achieved 99.89%, 84.18% and 81.38% accuracy respectively.**

I. Introduction

Communication is an essential skill to participate and live in a community. Hearing, speech, and vision are our primary senses that help us communicate with other people. However, sometimes people are either born without one of their senses or they lose one in their lives. As a solution to the problem people have invented ways to use one of the other senses for communication. Research in the field of deaf and mute is growing rapidly, especially on the alphabet sign recognition.

This project, focuses on the sign language recognition for enabling people to communicate with deaf and mute people. We implement three different algorithms to classify images of hand signs. Each sign indicates a letter from the alphabet.

In order for the algorithm to be able to detect the hand, image pre-processing needs to be done. The dataset we use has pre-processed images which helps us focus on classification.

Many classifiers have been used to solve this problem. These methods include Gaussian mixture model, naive Bayes classifier, support vector machine, k-nearest neighbors, decision tree and radial basis function classifiers.

In this project, we will be comparing three different classification methods. Then we will explain their performance and conclude into one of them in terms of performance and computational complexity. The algorithms we use are Convolutional Neural Network, Support Vector Machine, and K-Nearest Neighbor.

II. Description of the Data

American Sign Language (ASL) is a natural language that is used by the deaf and hard-of-hearing community in the United States and parts of Canada as a means of communication. It is a visually based language that mainly uses hand gestures. ASL has the same linguistic and different grammar properties as spoken English.

To achieve the sign language classification and to recognize the patterns of Sign language alphabets(SLA), we are using a highly constructed dataset called MNIST sign language dataset.

The MNIST (Modified National Institute of Standards and Technology database) Sign Language dataset is a variant of the original MNIST dataset that consists of images of hand gestures representing the letters of the American Sign Language alphabet. MNIST sign language dataset consists of a set of 27,455 cases of training images and 7172 cases of test images. Images in this dataset are of hands making different American Sign Language gestures rather than handwritten digits. All the images of the dataset represent a single 28x28 pixel (784 pixels) image with grayscale values between 0-255. the sign language MNIST is presented here and follows the same CSV format with labels and pixel values in single rows. The American Sign Language letter database of hand gestures represents a multi-class problem with 24 classes of letters (excluding J and Z which require motion). The MNIST data for sign language was obtained by considerably expanding the scan of the few (1704) color photos that were not clipped around the hand region of interest. An ImageMagick-based image pipeline was utilized to generate fresh data, and it includes hand-only cropping, gray-scaling, resizing, and at least 50+ variants to increase the quantity.

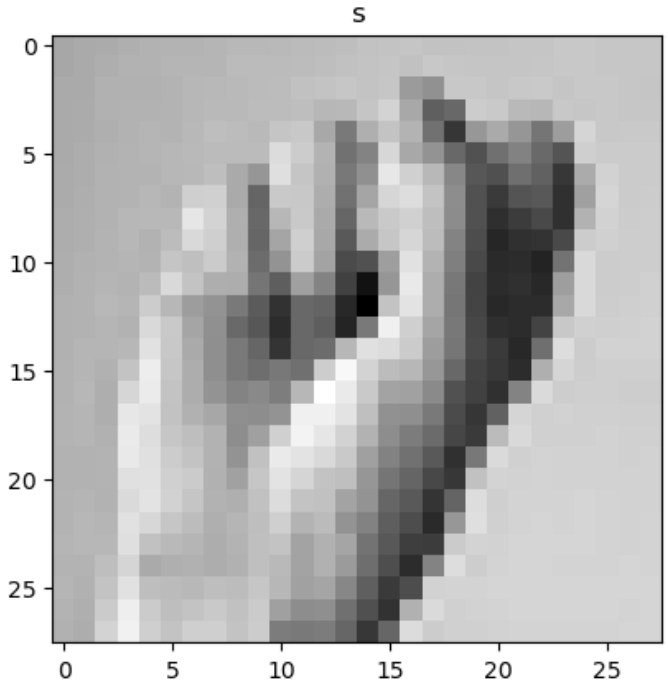

Fig. 1. Example image from dataset

Fig. 2. example images of MNIST dataset

III. Image Classification Models

A. Convolutional Neural Network

A convolutional neural network (CNN) is a type of artificial neural network built especially for processing and recognizing images. It is made up of several artificial neuronal layers that process and evaluate input images using a technique called convolution.

The ability of convolutional neural networks to automatically learn spatial hierarchies of features from input images makes them particularly helpful for image classification applications. This is done through the use of convolutional layers, which apply a series of filters to the input image to extract relevant features. These filters are often trained using supervised learning, in which the network is presented with labeled images and the corresponding labels and adjusts the filters to recognize the desired features. In addition to convolutional layers, CNNs also contain other types of layers such as pooling layers, which reduce the size of the input image and help to reduce overfitting, and fully connected layers, which perform classification on the features extracted by the convolutional layers.

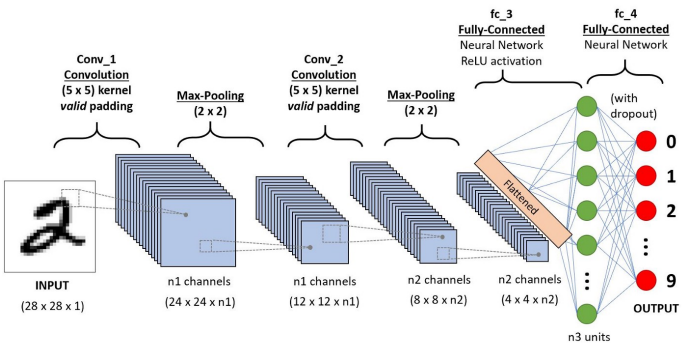

Fig. 3. layers of CNN model

In this project, we are going with a convolutional neural network because of its well-suited recognition features such as object detection and image classification. CNN will provide the best results when it comes to image classification and pattern recognition.

B. Support Vector Machine

Support vector machine is a supervised learning method which deals with the understanding and inferring from the function by labelling the training set of data. After performing hand pre-processing, edge detection and feature extraction the output values are classified into different classes. The algorithm first creates hyper planes to separates the data into two classes, any number of hyper planes can be constructed to classify the data. The approximate hyperplane can be represented as the one which offers largest separation or margin between any two classes. The hyperplane for which the margin is maximum is the optimal hyperplane. The hyperplane is measured in terms of distance present from the adjacent data point on where it is maximized on each side.

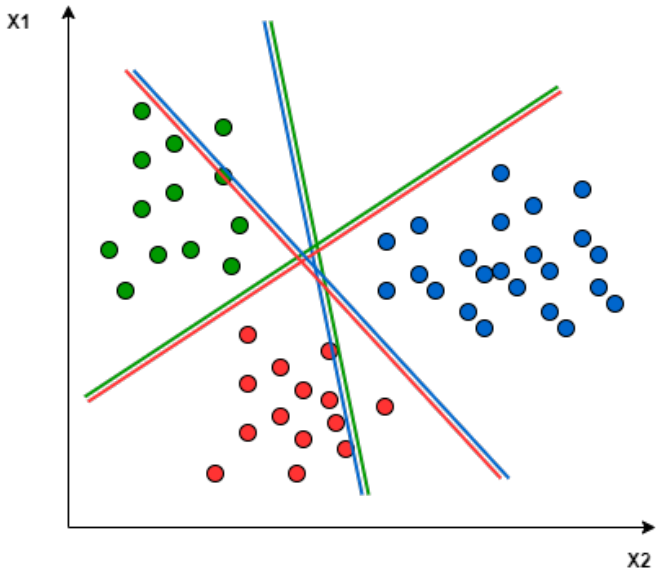

Fig. 4. Multiclass Support Vector Machine

C. K-Nearest Neighbor

K-nearest neighbor algorithm is a data classification method for estimating the likelihood that a data point will become a member of one group or another. It has shown remarkable performance on data with large example size. However, the performance of KNN can be affected by the number of neighbors and the distance measure. The number of neighbors can be any integer. Also, the metric we choose depends on the nature of the problem we try to solve. Some examples of distance metrics are Euclidean distance, Manhattan distance, Cosine distance, and Jaccard distance.

- Euclidean distance $d(x, y) = \sqrt{\Sigma_{i=1}^{n}(x_i - y_i)^2}$
- Manhattan distance $d = \Sigma_{i=1}^{n}|x_i - y_i|$

- Cosine distance $\cos\theta = \frac{\vec{a}\cdot\vec{b}}{||\vec{a}\cdot||\cdot||\vec{b}||}$
- Jaccard distance $J(A,B) = \frac{|A\cap B|}{|A|+|B|-|A\cup B|}$

The k-nearest neighbor algorithm has been regarded as one of the top algorithms due to its simplicity and low complexity. The training process consists of storing feature vectors and labels of the training images. In the classification process, the unlabeled query point is assigned to the label of its k-nearest neighbor.

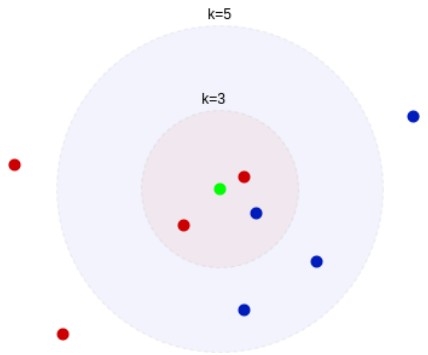

Fig. 5. K-Nearest Neighbor with K=3 and K=5

IV. Implementation and Validation

A. Convolutional Neural Network

Implementation of convolutional neural network for this project as done in the jupyter notebook using python3 and other libraries including Tensorflow, Pandas, Numpy, and matplotlib. We took the data from the MNIST train dataset which contain 27,455 data to train the model. At first, we supported the label from the data. The data for each image must then be normalized, and the labels must be encoded simply using a single hotkey.

This Convolutional Neural Network model consisted of 4 convolution layers (containing two 32 and two 64 filters), with each followed by a relu activation layer. For two convolution layers that have the same number of filters, we added one max pooling layer. A dropout layer was then added to help prevent overfitting. After this, the data was flattened and fed to a fully-connected dense layer of 556 and 128 nodes with relu activation. Finally, with one more dropout layer the model ended with a 26-node dense layer with softmax activation. Because of the integer target model we compiled it with categorical crossentropy loss. additionally, we implemented 3 epochs to the model and we were able to get predictions that are remarkably good ever since the first epoch.

For the validation part, we took the testing dataset which contains 7172 data, and compiled it with the CNN model. The validation was accurate and the model is able to predict the alphabet from the MNIST sign language images.

B. Support Vector Machine

First we start with Support Vector Machine algorithm, and since the this is a multiclass problem, I chose one vs rest decision function. The script is pretty straight forward, we use all of our training data to estimate the parameters for our SVM model and afterwards validate it with the test dataset. The more data we use the more accurate our model will become. In this case, our classifier manages to predict the test images at an 84% accuracy

C. K-Nearest Neighbor

The implementation of K-Nearest Neighbors is done in Jupyter Notebook using python3. The libraries used are Numpy, Pandas, and Matplotlib. The algorithm is implemented from scratch using two different distance metrics. Training happens using an intel i7 10th gen CPU. First, the distance metrics are implemented (Euclidean distance and Manhattan distance) then standard scaler is implemented to normalize the data. Lastly, a class for the main K-nearest neighbor algorithm follows.

In order to observe the performance of different parameters, the algorithm is trained over two different data sizes and 4 different K values. First, 10000 data samples are used to train the model over k values 3, 5, 7, 9 for each distance metric. Then, 27455 data samples are used to train the model over k values 3, 5, 7, 9 for each distance metric.

Iteration	**KNN**		
	Dataset size	***K value***	***Distance Metric***
1	10000	[3,5,7,9]	Manhattan
2	27455	[3,5,7,9]	Manhattan
3	10000	[3,5,7,9]	Euclidean
4	27455	[3,5,7,9]	Euclidean

V. Classification Results

A. Convolutional Neural Network

The Convolutional Neural Network model which we implemented to this dataset gave us remarkably good results. When we trained the data set with the training data by applying three epochs we got the result of 100 percent accuracy in prediction. While validating the model with the MNIST test data we were able to visualize the result with the respective alphabets.

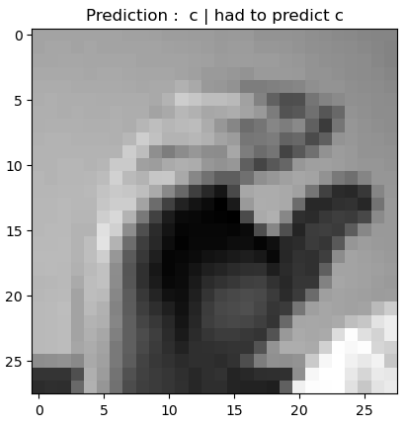

Fig. 6. Prediction is same as sign language

B. Support Vector Machine

Sign minst data is consider to train and test the SVM model. The expected result of the systme is s successfully. The SVM algorithm achieved good performance compared to KNN. f1 score : 0.8418851087562744 accuracy score : 0.8418851087562744 In this way we got 84 percent.

C. K-Nearest Neighbor

Different KNN models are trained with different datasizes and k values to observe the performance using different parameters.

TABLE I
RESULTS

	KNN		
Dataset size	***K value***	***Distance Metric***	***Accuracy***
10000	[3]	Manhattan	0.034 %
10000	[5]	Manhattan	0.034 %
10000	[7]	Manhattan	0.034 %
10000	[9]	Manhattan	0.034 %
27455	[3]	Manhattan	0.034 %
27455	[5]	Manhattan	0.034 %
27455	[7]	Manhattan	0.034 %
27455	[9]	Manhattan	0.034 %
10000	[3]	Euclidean	77.80 %
10000	[5]	Euclidean	76.75 %
10000	[7]	Euclidean	74.94 %
10000	[9]	Euclidean	74.15 %
27455	[3]	Euclidean	81.38 %
27455	[5]	Euclidean	81.30 %
27455	[7]	Euclidean	81.20 %
27455	[9]	Euclidean	80.68 %

From the table above we can see that the distance metric has a significant effect into the accuracy of KNN. Manhattan distance does not give us good accuracy under any dataset size or k value. More specifically, it gives us 0.034 % for all data sizes and k values. On the other hand, Euclidean distance for data size of 10000 samples and k value of 3 gives us 77 % accuracy. When we increase the data size to 27455 images we get better accuracy. The best k value, datasize, and accuracy are 3, 27455, and 81.38 % respectively.

VI. CONCLUSION

In this project we implemented 3 models (SVM,KNN,CNN). The experiment results are encouraging CNN gives the best results with this dataset compared to other machine learning models with an accuracy of 100 percent. and SVM is much more efficient than KNN in predicting from the given test dataset. We got an accuracy of 84 percent for SVM and 80 percent for KNN. The future work of the ASL Recognition system can be more versatile working on the Mobile application and building a language translator system that involves all the sign language dictionaries of the different nations to help mute people to communicate easily across the world. Other machine learning applications and tools like OpenCV can be implemented to make ideas to bring into real-world applications.

CONTRIBUTION

The team communicated physically and virtually about the project ideas and updated all shared files as progress was made. We worked together for researching different papers and come up with this sign language recognition idea. To implement this project with we found this particular data from Kaggle website and agreed to proceed with that in one of our meetings. Completing the report was a collaborative effort where everyone took part in writing and proofreading. Each person in the group come up with one machine learning model to check which one works better for this dataset. Chinmay implemented the CNN model, Spyros implemented the KNN model and Suyog comeup with the SVM for the same dataset which we discussed earlier.

RESOURCES

Dataset used in this project: https://www.kaggle.com/datasets/datamunge/sign-language-mnist

REFERENCES

[1] Bilgin, Metin & Mutludoğan, Korhan. (2019). American Sign Language Character Recognition with Capsule Networks. 10.1109/ISMSIT.2019.8932829.

[2] Zhang, Shichao & Li, Xuelong & Zong, Ming & Zhu, Xiaofeng Cheng, Debo. (2017). Learning k for kNN Classification. ACM Transactions on Intelligent Systems and Technology. 8. 1-19. 10.1145/2990508.

[3] Utaminingrum, Fitri, I. Komang Somawirata and Gilbert Dany Naviri. "Alphabet Sign Language Recognition Using K-Nearest Neighbor Optimization." J. Comput. 14 (2019): 63-70.

[4] T. Makkar, Y. Kumar, A. K. Dubey, Á. Rocha and A. Goyal, "Analogizing time complexity of KNN and CNN in recognizing handwritten digits," 2017 Fourth International Conference on Image Information Processing (ICIIP), 2017, pp. 1-6, doi: 10.1109/ICIIP.2017.8313707.