A Comparative Analysis of Classification Algorithms for American Sign Language Recognition

Pravesh Koirala
KFSCIS
6307146
pkoir002@fiu.edu

Behroze Chaudhary
KFSCIS
5320860
bchou004@fiu.edu

Ibrahim Alyami
KFSCIS
6305315
ialya001@fiu.edu

David Acosta
KFSCIS
6056312
dacos062@fiu.edu

***Abstract*—As a part of our class project (CAP5602), we present a comparative analysis of four different Machine Learning models on the Sign Language MNIST dataset [1]. The models we use are of varying complexities ranging from Logistic Regression, a simple linear classifier to Convolutional Neural Network (CNN), the current state of the art in Image Recognition, achieving a maximum accuracy of 97%. The results we obtain reaffirms the superiority of Deep Learning models in unstructured pattern recognition problems. We perform ablation study on Dropout and Batch Normalization method for our CNN model and demonstrate that they indeed aid in the increase of the accuracy. Lastly, we also use additional classification metrics such as macro-precision, macro-recall, and macro-f1 score to benchmark our models.**

***Index Terms*—project, Convolutional Neural network, support vector machine, classification, logistic regression, multilayer perceptron, sign language**

I. Introduction

Sign languages are a class of language primarily used by people who are deaf and hard of hearing. They are distinct from spoken languages in the sense that they do not use auditory signals (i.e. speech) and instead rely on hand signs, gestures, and facial expressions for communication. Sign languages differ from country to country and culture to culture. Within the US, the most popular sign language in use is the American Sign Language, or the ASL. The beginning of ASL is not clear but it is speculated that ASL arose some 200 years ago from the intermixing of some local sign languages and the French Sign Language [2]. ASL, in itself is a complete language with its own linguistic properties and grammars and it can also be translated to other languages [5]. It is generally believed that more than 500,000 people use the American Sign Language [3].

Since ASL enjoys such a large popularity in a significant cluster of American population, it is desirable that it be accessible to other people as well in order for a smooth communication with those who are deaf and hard of hearing. Unfortunately, people who do not have disability related to hearing rarely teach themselves the ASL except to communicate with their immediate family members. Therefore, there has been multiple attempts to create an automatic sign language identification system to translate the hand gestures to English or equivalent languages [10]. While we discuss many of these attempts in depth in the section Related Works, most of them employ either hand glove based method or an image recognition based method for sign language identification. In this paper, we explore the latter i.e. image recognition based method for the same.

Image Recognition (IR) is a supervised Machine Learning problem which essentially involves automatic recognition (usually classification) of image artifacts. IR has been a vastly researched field which is too comprehensive to be summarized here. [4] provides a good introduction to this field. However, there are two classes of methods usually employed for IR: Feature-based, and End-to-End. The first i.e. Feature-based method involves extracting image features such as Histograms of Oriented Gradients (HOGs), Scale Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), etc. [4] and using supervised classification algorithms such as Support Vector Machines (SVM), Logistic Regression, Random Forests etc. on the extracted features. A more recent trend in IR is the usage of End-to-End models and methods such as Neural Networks. Convolutional Neural Networks (CNNs) in particular have been shown to produce outstanding results in comparison to feature based methods [8], with newer CNN based architectures taking the lead in recent benchmarks [9].

In this paper, we use four supervised Machine Learning classifiers i.e. Logistic Regression, Support Vector Machines, Multi Layer Perceptron, and Convolutional Neural Networks to classify ASL gestures using a publicly available dataset i.e. Sign Language MNIST [1]. We do a comparative study of all of these methods on our dataset and draw an important conclusion in regards to the efficacy of different kinds of models in Image Classification problem.

1) Extra Credits: In order to qualify for extra-credits, we have used three novel (not covered in class) methods in this project.The first one is Batch Normalization, which is a technique which normalizes the output of a particular layer in order to facilitate stability and faster convergence while training the network [6]. Similarly, the second one i.e. Dropout is a technique where a select portion of a particular NN layer is randomly removed during training. It is a popular method of regularization which improves the generalization capabilities to the model [7]. Lastly, we use metrics other than accuracy, such as Precision, Recall, and F1 to better compare our model's classification performance.

II. Related Works

There have been many attempts on automatic identification of Sign Languages (SL). The first attempt at automatic sign language recognition was done by Tamura and Kawasaki [20]. They used color segmentation methods to recognize Japanese SL on the basis of the geometry, movement and position of the hand. Starner et. al. came up with two models based on the Hidden Markov Model (HMM) algorithm and one of them achieved up to 98 percent accuracy using a camera mounted on a cap on a vocabulary of 40 words [19]. Liang and Ouhyoung also developed a real time gesture recognition model using a specialized glove in Taiwanese Sign Language using HMMs [11]. They reported a continuous recognition rate of 80.4% for a vocabulary of 240 words. Yang et. al used a time-delayed Neural Network to learn motion patterns of hand gestures of the American Sign Language [15]. They extract the features using Affine Transform and report a final accuracy of 99.02% over a vocabulary of 40 words.

In recent years, the rise of Deep Learning has seen newer models that have attempted to improve over the conventional methods and results. Koller et. al. developed "Deep Sign," which was a hybrid of Convolutional Neural Networks (CNNs) and HMMs [14]. They used the image recognition capabilities of the CNN and leveraged the sequence recognition capabilities of the HMM on three benchmarks i.e. SIGNUM, RWTH-PHOENIX-Weather 2012, and RWTH-PHOENIX-Weather-MULTISIGNER 2014 to significantly improve the existing results. Similarly, Koller et. al. used CNNs with Long Short-Term Memory (LSTM) in their work "Re-Sign" to outperform the state of the art by 10% absolute and 30% relative accuracy [18]. [10] presents a Survey of state of the art in Sign Language Recognition which readers can make use of to acquaint themselves with the comprehensive literature.

Apart from the general literature on Sign Language detection, there also has been multiple works that make use of this very dataset. Luqman et. al. use Gabor filter and CNN to achieve an accuracy of 99.90% [16]. Their approach makes use of both the raw image as well as the Gabor transformed images which they use to train the CNN model. Goswami and Javaji also use a CNN based approach and report an accuracy of 99% [12]. Interestingly, they use BatchNormalization in their CNN but do not use Dropouts. Xiao et. al. make use of the Capsule Neural Network to achieve accuracy of 99.6% [21]. Similarly, Rathi uses a Transfer Learning approach where he retrained two pre-trained models i.e. Inception_V3 and MobileNet on this dataset, and achieved a result of 95.06% and 93.36% [17].

III. Data and Methods

A. Data

The dataset we have used is the Sign Language MNIST image dataset [1] which is freely available on the online Machine Learning community Kaggle. This dataset consists of 27,255 training points and 7,172 testing points. These data points are in the form of gesture images for each of the English Alphabets (24 classes of letters) excluding J and Z, because they require motions instead of a stationary hand symbol and thus are not suitable for an image-recognition task. Each of the data points i.e. images are constructed using 28x28 pixels and have a total 784 pixels/sample. The data has been made available in the CSV format so the labels and pixel values are in single rows with each entry representing a grayscale pixel value between 0-255.

The images are pre-processed using techniques such as cropping, gray-scaling, resizing, and using an image processing pipeline to modulate the data. The original number of images for this data was 1704. The rest were generated by augmenting the images using modification and expansion strategies like filters ('Mitchell', 'Robidoux', 'Catrom', 'Spline', 'Hermite'), along with 5% random pixelation, +/- 15% brightness/contrast, and finally 3 degrees rotation.

B. Methods

We used four methods to compare image classification performance against the Sign Language MNIST dataset, namely Logistic Regression, Support Vector Machine (SVM), Multilayer Perceptron (MLP), and Convolutional Neural Networks(CNNs). In the following paragraphs, we will briefly explain the parameters that we used in each model.

We trained our Logistic Regression model for multi-class classification using the multinomial loss function (also known as the cross entropy loss). We used L2 regularization while training this model in order to reduce possible over-fitting. The training iteration field was left default i.e. a value of 100 while all of the classes were given an equal weight. This model was trained using the Large-scale Bound-constrained Optimization i.e. LBFGS algorithm.

Secondly, we used SVMs. We applied two different types of mathematical functions (kernels) in our SVM implementation. We trained our models with the Radial Basis Function (RBF) kernel and the linear kernel. The parameter C, also known as the Regularization parameter is shared by all SVM kernels. It provides a mechanism to balance the risk of misclassification of training samples against the decision surface's simplicity. Gamma, also known as the kernel coefficient refers to the amount of effect that a single training example has [26]. In our experiments with the linear kernel, we used the value of regularization parameter as 1. And we used the default value for our kernel coefficients which calculates the value of gamma as following:

$$gamma = \frac{1}{|X| * var(X)}$$

(where X is our input vector and var is the variance function)

In the Multilayer Perceptron (MLP) model, we used three hidden layers and one output layer. The three hidden layers were of the size 500, 400, and 300, whereas the output layer had 24 nodes representing each of the output class. We trained our model using the Rectified Linear Unit (ReLU) activation function for each of our hidden layers and used Softmax activation for the last layer. The entire network was trained

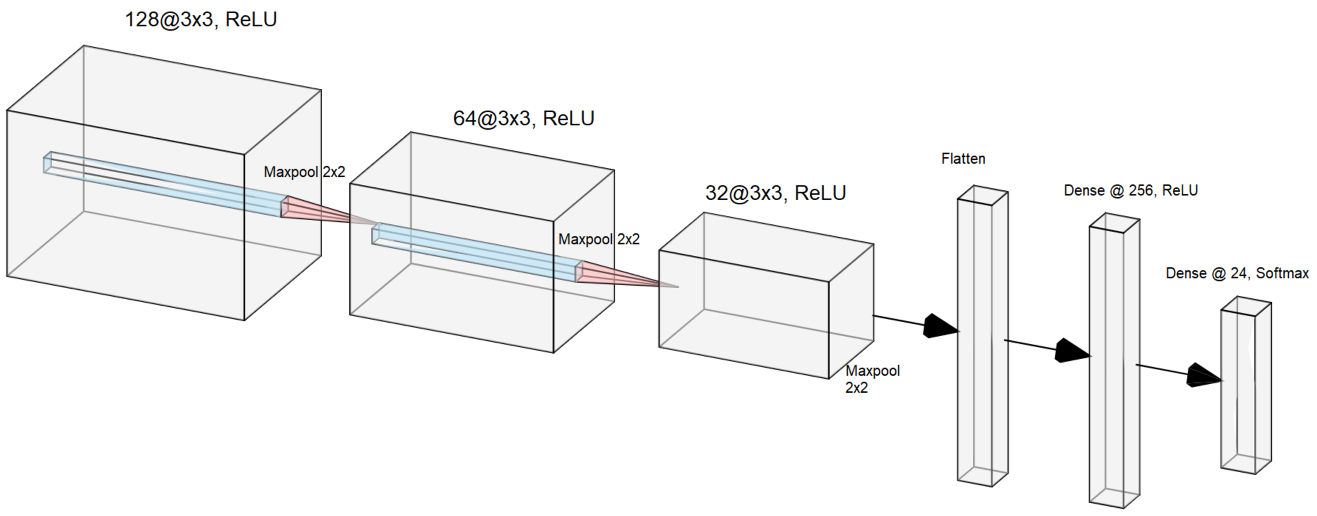

Fig. 1. A schematic of our vanilla CNN model (Dropouts and BatchNorm are not shown). The figure was generated using *NN-SVG*

.

on the cross entropy loss function using the ADAM [27] optimization algorithm.

Finally, for the Convolutional neural network (CNN) model, we used three convolution layers with 128, 64, and 32 filters of size 3x3. All of these filters had the "same" padding property i.e. they did not change the size of their input. Max pooling was used to downsize the layers at each step. Since we also did an ablation study on Dropout and BatchNorm, we constructed two more similar architectures with the only difference being the addition of these two layers. For each of these cases, we used Dropout with the parameter 0.2 which meant that our model excluded 20% of the neurons while training. After the convolution layers, all of our architectures flattened their layers and used an additional dense layer of 256 neurons followed by an output layer of 24 neurons. All of the layers except for the output layer used the ReLU activation, whereas the output layer used softmax. Figure 1 describes our basic architecture.

IV. Experiments

We conducted detailed experiments using four different kinds of algorithms. We used Logistic Regression as a basic benchmark, Support Vector Machines with both linear and non-linear kernels, Multi-Layer Perceptron, and Convolutional Neural Networks. Our experiments were run on Google Colab notebooks as it provided GPUs for some of our complex models. For models such as Logistic Regression, MLP, and SVM, we used a popular Machine Learning framework in the python language ecosystem i.e. Sci-Kit [22]. For our CNN model, we used Keras with Tensorflow backend [23]. To load the csv file and perform basic data cleanup and array conversion, we used the Pandas library [24], whereas all other numeric manipulations were done using NumPy [25].

To make our experiments reproducible, we set a random seed with a value of 10 at the start of each of our experiments. While the standard practice suggests that we repeat our experiments multiple times and report the average, due to our time constraints we opted for a single run of our models. To better evaluate our models, we analysed not only the accuracy of the setup, but also used other classification metrics such as Precision, Recall, and F1. The results of our experiments are tabulated in the table I.

A. Ablation Study

For one of our model i.e. Convolutional Neural Network, which is a Deep Learning model, we attempted different kinds of architectures to see the effects on the overall accuracy of our classification task. In particular, we first tried CNN as-is without any regularization or normalization. Next, we added Batch Normalization on the layers of the network. Afterwards, we added both Batch Normalization and Dropouts. By sequentially adding these elements to our architecture, we performed an Ablation Study, the results of which are tabulated in I.

TABLE I
Results (rounded for readability)

Methods	**Accuracy**	**Precision**	**Recall**	**F1**
Logistic Regression	0.67	0.66	0.66	0.66
SVM (Linear)	0.78	0.77	0.77	0.76
SVM (RBF)	0.84	0.83	0.83	0.83
Multi Layer Perceptron	0.81	0.8	0.8	0.79
CNN	0.93	0.93	0.92	0.92
CNN + BatchNorm (BN)	0.94	0.94	0.94	0.94
CNN + BN + Dropout	**0.97**	**0.97**	**0.97**	**0.97**

V. Discussion and Conclusion

We have demonstrated the efficacy of multiple classifiers on a publicly available image dataset. While the accuracy that we have obtained is certainly below the state-of-the-art but our methods have achieved a modest performance that proves our ability of applying different ML algorithms to novel datasets. Among our models, logistic regression, which is the simplest possible linear model, achieves an accuracy of 67% which is better than the random classification error of 4.16%. This suggests that our dataset has plenty of linear features that can be leveraged by other non-linear classifiers to achieve an accuracy which is at least better than that.

Similarly, Support Vector Machine with RBF kernel achieves a comparable performance to that of a Multi-Layer Perceptron which demonstrates its versatile ability as a non-linear classifier against a powerful feature-learning classifier such as the MLP. This leads us to believe that combining SVM with image feature extraction techniques such as SIFT and SURF would surely improve the performance of the SVM. This is an avenue that we believe should be explored further.

Finally, it does not come off as surprising to us that the best results were obtained using CNN which is the state-of-the-art algorithm for Image Recognition. However, our ablation study does prove that using BatchNorm and Dropout significantly improve the accuracy of our model both while used independently as well as when used in conjunction.

Acknowledgment

We take this opportunity to express our gratitude towards our professor Dr. Cuong Nguyen whose illuminating lectures, easy to follow examples, and a hands on approach to teaching has proved immensely beneficial to our understanding of Artificial Intelligence. We also thank our TAs and classmates for making CAP5602 such an enjoyable experience. Finally, team members of group 1 would like to appreciate each other for such a well-executed collaboration.

References

[1] “Sign Language MNIST.” https://kaggle.com/datamunge/sign-language-mnist (accessed Nov. 20, 2021).

[2] [redacted]

[redacted]