Military Hand Signals Classification using Deep Learning

Mohit Sai Aravind N[1][0000-0002-4230-7292], Hariharan S[2][0000-0002-3620-7068], and Ayesha Shaik[3][0000-0002-9804-8031]

[1] Vellore Institute of Technology, Chennai
mohith01@gmail.com
[2] Vellore Institute of Technology, Chennai
hariharanmay21@gmail.com
[3] Vellore Institute of Technology, Chennai
anoorcse@gmail.com

Abstract. Military Hand signals are a method of visual communication for field use and are now the most common forms of communication during operations. It is important that communications in missions are clear, distinct and understandable. Hand Sign Language (SL), have become a standard part of communication in the military, especially when voice communication is not desirable or silence has to be maintained for security. With the advent of drones and high-precision cameras, communication using hand signals can be very effective, as it is quick and the support device does not need to be in proximity. A trained human operator may be required to translate the meaning of these hand signals. Instead, a machine learning algorithm can be developed to recognize and translate these messages, saving time for translation and the requirement of specialized training for the interpreters. This project aims to create a method that can translate army hand signals into their respective labels, which can be used by drones for reconnaissance and safety. Our model, which is based on Convolution Neural Networks, achieves a cross-validation accuracy of 98.32% and test accuracy of 97.94%.

Keywords: Convolutional neural network · Computer Vision · Sign language recognition · Military Applications.

1 Introduction

Military hand signals are used as a means of non-verbal communication, usually to describe key commands to be used in an operation. These signals can be static or gesture-based[1]. Signals such as numbers, ok sign, thumbs up, and down are static signals as the command can be recognized by looking at the hand of the user alone. Other dynamic gestures may be used such as rotation of arms in upwards, downward direction to signal different commands. A mixture of both may be used to communicate the severity/urgency of the message.

Militaries use different encodings and hand signals for short distance communication. For our project, we have used a subset of the hand signals used by the USA army.

There are various situations in military operations where remote monitoring may be required. Hostage situations, rescue operations, disaster control, etc use UAVs to monitor and pass commands to the field agents. UAV's mounted with cameras can detect signals from medium-range distances. Translation of these commands is usually done by a physical operator. However, this process of translation can be automated by various approaches, including hardware-assisted, software based methods. Hardware-assisted methods involve the operator using specialized gloves with gesture detection sensors and transmitter systems for communication . Purely software solutions will involve scanning an image or video input and making predictions for the expected signals based on the same.

This project aims to help in recognising the hand signals used by the army using an trained deep learning model (Convolutional Neural Network) as a software solution. The project demonstrates the translation of a subset of static hand signals used by NATO military manual.[1]

2 Literature Survey

There are a plethora of sign language translation and recognition models in the field. Works on SL recognition has used various languages like the American SL [8,2], Korean SL[10], Chinese SL[12], and Japanese SL[13]. C. Wang, W. Gao, and J.Ma explain Hidden Markov model based recognition system for Chinese SL which has a large-scale set of more than 5000 signs as its vocabulary[11]. Christian Vogler and Dimitris Metaxas develop scalable hand recognition systems which solves challenges such as handling simultaneous events, like signals which involve simultaneous change in hand movement and shape[8]. They train Hidden Markov Models to recognise the phonemes, instead of whole signs, and it can help in recognition of large-scale vocabularies. J. S. Kim, W. Jang et al present a model which uses pair of data-gloves as sensing device for detecting motions of hands and fingers. They are able to recognise Korean SL (KSL) and provide a translation for the same in text format in Korean Language[10].

Convolutional neural networks (based on [16]) are deep learning algorithms that can learn features by assigning weights and biases to image-based inputs. These models are very successful at image recognition. [17,18] For Image manipulation, we have applied Gaussian blurring as they can make the image a bit more robust and it makes the model stabilized to noise. There are some projects that prepare data using Xbox Kinect Software to capture images. Kinect helps us in creating models with better accuracy for ASL as they are able to learn depth-based features[2]

In L Pigou paper [3], they recognise 20 Italian gestures using convolutional neural networks. Their model achieves an accuracy of 91.7%. For the task of gesture spotting in ChaLearn competition, the model gives a Jaccard Index value of 0.789. In our model, we have used batch normalization as explained in [4] how Batch Normalization makes the optimization landscape more smoother allowing for faster training.

The human hands and arm signals are recognized using a computer system according to the paper published by Lampton and his colleagues[7] which consists of two video cameras and a system to recognize gestures by analyzing the position and movement of the hands. The recognition rate, however at best is 87% for a individual and the minimum accuracy was 57%. They mainly make use of the Cybernet Gesture Recognition Software (GRS) for tracking the movements. They have used the sensors and mapped the hand and arm gestures with a virtual environment so that soldiers can train using these simulations, whereas in our model, we design a CNN which can be used with drones for reconnaissance and similar purposes. Different ways of augmenting datas such as using transformations, Generative Adversarial Networks (GANs) and other approaches are explained in this paper[6]. To increase the images in our dataset, we have mainly used transformations like rotations, reflection and scaling.

3 Methodology

3.1 Data

We combine some of the signs from Sign Language MNIST Dataset[2] in Kaggle with another custom dataset. From the Kaggle dataset, the signs in Figure 1 is used.

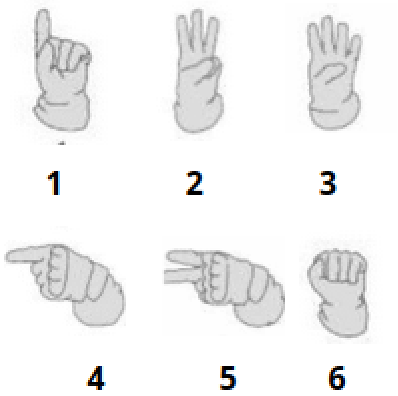


Fig. 1: This are the signs used for this project and all of them are signals used by the army.

After these signs are chosen, a custom dataset is built by capturing images of these signs using opencv-mouse bindings and is stored into their respective sign folders.

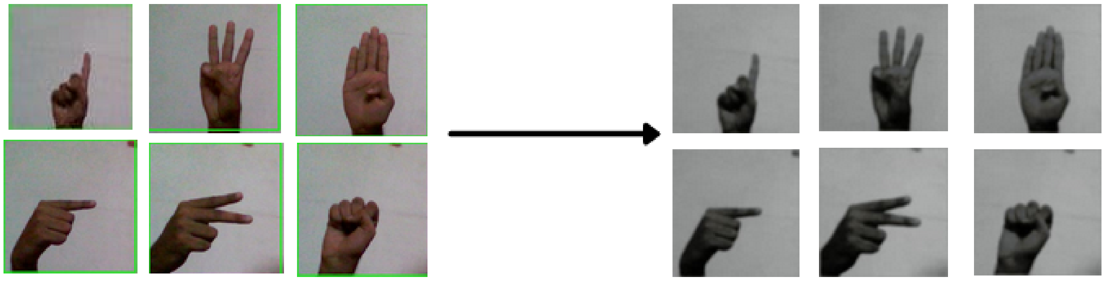


Fig. 2: Normal image to Gaussian blurred grayscale image

These captures images are converted from RGB to grayscale and blurred using Gaussian blur as shown in Figure 2. Gaussian blur is extensively used in image processing applications, mainly to handle noise and reduce/ standardize the details in the images. Mathematically, we apply the convolution of the image with the Gaussian function.

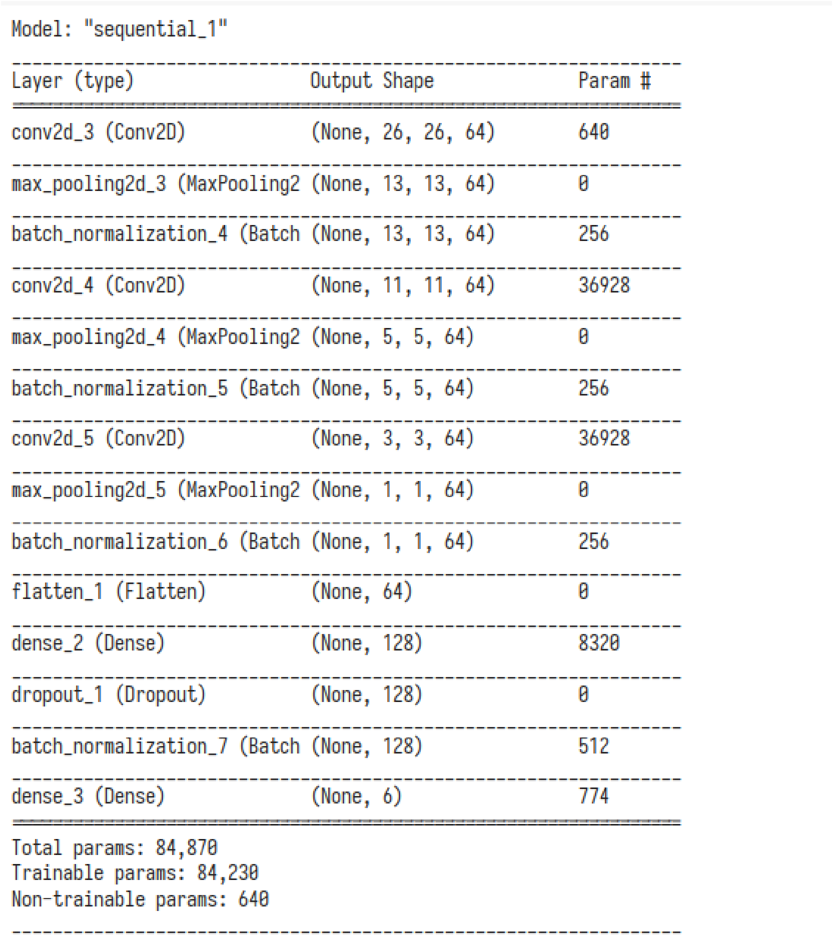
$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x)^2/2\sigma^2} \tag{1}$$

Equation 1 represents a typical Gaussian function $G(x)$ for normal distributions in one dimension. Parameter σ is the standard deviation of the distribution, and x represents the value of the variable, which is the distance of the point from origin along the horizontal axis.

Then, all these images are resized into shape of 28x28 image so that model will be able to train faster on these images. A total of 6,558 images are generated using a script and combined with the kaggle dataset to form 13,129 images and their respective labels are one hot encoded.

3.2 Proposed Architecture

We use three Conv2D layers for feature extraction, followed by Flatten, then followed by the dense layers, which help in classification, and can handle one-dimensional data. Each of the Conv2D layers have a kernel size of (3,3) and uses the Relu [18,19] activation. These Conv2D layers is followed by Max pooling which uses max-pooling of the 2D dimensions with kernel size of (2,2) for better accuracy. Convolution 2D layers work best for images[3]. The output from these convolutional layers are flattened and passed to the dense layers which help in classifying the image.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 64)        640
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 13, 13, 64)        0
_________________________________________________________________
batch_normalization_4 (Batch (None, 13, 13, 64)        256
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 11, 11, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
batch_normalization_5 (Batch (None, 5, 5, 64)          256
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
batch_normalization_6 (Batch (None, 1, 1, 64)          256
_________________________________________________________________
flatten_1 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               8320
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
batch_normalization_7 (Batch (None, 128)               512
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 774
=================================================================
Total params: 84,870
Trainable params: 84,230
Non-trainable params: 640
_________________________________________________________________
```

Fig. 3: Model Summary

The summary of the model is shown in Figure 3. Convolution and max-pooling layers are stacked with batch normalization layers which applies a transformation that such that the output and standard deviation is close to 0 and 1 respectively. It enhances model performance and reduces training time[4]. Dropout technique is used to prevent overfitting by reducing the capacity of the neural network. Dropout techniques achieves this by randomly ignoring certain neurons while training.

3.3 Generalization and Training

We have split the data into training, cross-validation and test set with a ratio of 80:10:10. Methods such as dropout[5] and data augmentation[6] have been applied during training to control over fitting. The data augmentation is performed in real time and consists of zooming up to 10%, shifting images horizontally and vertically 10%, and rotations up to 20 degrees in both the directions. The model is compiled using Adam optimizer with learning rate as 0.001 and categorical cross entropy is the loss function that has been used. Training is done for 50 epochs and the hyper parameters are tuned in such a way that it performs well on the development set. The entire training is done in GPU based environment on google colab using the Python library Tensorflow.

4 Results

The model on training with keras achieves a train accuracy of 95.12% and validation accuracy of 98.32% after training with 50 epochs. On evaluating the model with test dataset, it gives an test accuracy of 97.94%. The predictions for test images along with the confidence of prediction can be displayed as follows:

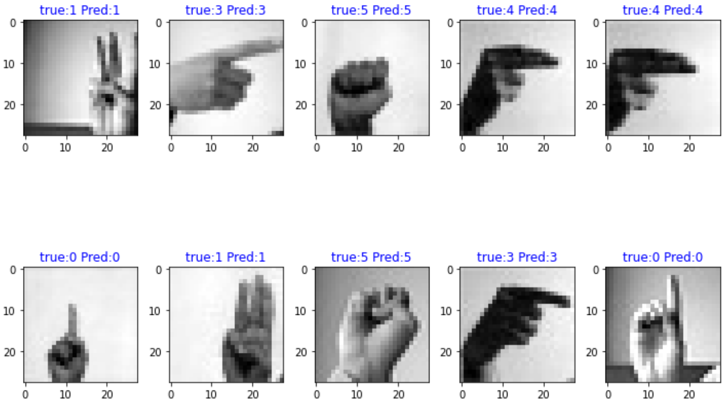
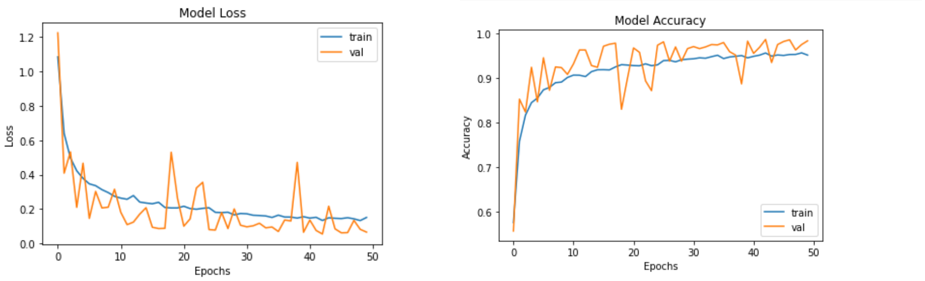


Fig. 4: Test images with predicted and real labels

Some of the samples on the test set with the predicted labels (results of the model) and true labels are shown in the Figure 4. The dropout and image augmentation prevents the model from overfitting.

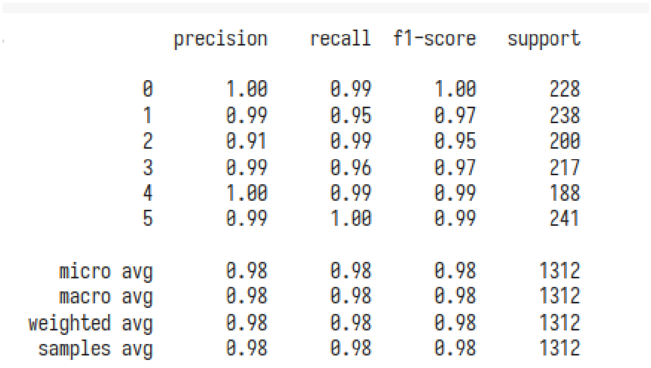


(a) Model loss vs epochs

(b) Model accuracy vs Epochs

Fig. 5: Plots of model loss and accuracy while training the model

The graphs of model loss and model accuracy with each epoch is plotted in Figure 5. From the graph, we can see that the model does not overfit the data during the training.



```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       228
           1       0.99      0.95      0.97       238
           2       0.91      0.99      0.95       200
           3       0.99      0.96      0.97       217
           4       1.00      0.99      0.99       188
           5       0.99      1.00      0.99       241

   micro avg       0.98      0.98      0.98      1312
   macro avg       0.98      0.98      0.98      1312
weighted avg       0.98      0.98      0.98      1312
 samples avg       0.98      0.98      0.98      1312
```

Fig. 6: Precision and Recall Table

Metrics such as precision, recall and F1-score for the task of classifying the hand signals is shown in Figure 6.

Precision is the ratio between the actual positive results and the total predicted positive results for a label.

$$Precision = \frac{TruePositives}{TruePositive + FalsePositive} \tag{2}$$

Recall is the ratio between the actual positive results and the total actual positive results for a label.

$$Recall = \frac{TruePositives}{TruePositive + FalseNegative} \tag{3}$$

F1-score is the harmonic mean of precision and recall.

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

Support is the number of actual occurrences of the label.

The classification results of the model gives a high precision and recall score ranging from 0.91 to 1.0.

5 Conclusion and Future Work

Hand signals are extensively used for non-verbal communication in militaries. This work shows how trained models can be used to recognise different signs of army signals not included in the training set. This project classifes a subset of the static hand signals used in the NATO army manual. Thus, we can predict static military hand signals with high accuracy using convolutional neural networks. Future work may involve extending this solution to include dynamic gesture based predictions using Hidden Markov Models. Hand signal detection can be extended to other sign datasets like fire rescue and search operations. Background subtraction can be added to further optimize the model. Improved speed and durability to noise for complex gestures can be fine-tuned.

References

1. Training Circular 3-21.60 Visual Signals (PDF), 5 May 2020
2. Sign Language MNIST For Hand Recognition Tasks, 2019
3. Pigou, Lionel, et al. ”Sign language recognition using convolutional neural networks.” European Conference on Computer Vision. Springer, Cham, 2014.
4. Santurkar, Shibani, et al. ”How does batch normalization help optimization?.” Proceedings of the 32nd international conference on neural information processing systems. 2018.
5. Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting.” The journal of machine learning research 15.1 (2014): 1929-1958.
6. Mikołajczyk, Agnieszka, and Michał Grochowski. ”Data augmentation for improving deep learning in image classification problem.” 2018 international interdisciplinary PhD workshop (IIPhDW). IEEE, 2018.
7. Lampton, Donald R., et al. Gesture Recognition System for Hand and Arm Signals. ARMY RESEARCH INST FOR THE BEHAVIORAL AND SOCIAL SCIENCES ALEXANDRIA VA, 2002.
8. C. Vogler and D. Metaxas. Handshapes and movements: Multiple-channel american sign language recognition. In Springer Lecture notes in Artificial Intelligence, volume 2915, pages 247–258, January 2004.
9. T. Starner and A. Pentland. Visual recognition of American sign language using hidden markov models. In Proceedings of the International Workshop on Automatic Face and Gesture Recognition, 1995.
10. J. S. Kim, W. Jang, and Z. Bien. A dynamic gesture recognition system for the Korean sign language KSL. IEEE Transactions on Systems, Man and Cybernetics, 26(2):354–359, 1996

11. C. Wang, W. Gao, and J. Ma. A real-time large vocabulary recognition system for Chinese Sign Language. In I. Wachsmuth and T. Sowa, editors, Lecture Notes in Artificial Intelligence, volume 2298, pages 86–95. Springer, 2002.
12. W. Gao, G. Fang, D. Zhao, and Y. Chen. Transition movement models for large vocabulary continuous sign language recognition (csl). In Sixth IEEE International Conference on Automatic Face and Gesture Recognition, pages 553–558, 2004.
13. H. Sagawa and M. Takeuchi. A method for recognizing a sequence of sign language words represented in a japanese sign language sentence. In Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition, pages 434–439, Grenoble, France, March 2000.
14. B. Bauer, H. Hienz, and K. Kraiss. Video-based continuous sign language recognition using statistical methods. In Proceedings of the 15th International Conference on Pattern Recognition, volume 2, pages 463–466, September 2000
15. Chai, X., Li, G., Lin, Y., Xu, Z., Tang, Y., Chen, X., Zhou, M.: Sign Language Recognition and Translation with Kinect (2013). Language Recognition and Translation with Kinect.pdf
16. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11) (1998)
17. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. Advances in Neural Information, 1–9 (2012).
18. Schmidhuber, Jurgen, U. Meier, and D. Ciresan. "Multi-column deep neural networks for image classification." 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2012.
19. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML 2010), pp. 807–814 (2010)
20. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier networks. In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics 15, pp. 315–323 (2011).