Comparison of Neural Network Models for Automatic Sign Language Identification

Prithwish Ganguly

Ramakrishna Mission Vivekananda Centenary College, Rahara

American Sign Language (ASL)

- A complete natural language with similar linguistics as a spoken language.
- Characterized by the movement of hands and faces.
- All English alphabets can be represented by a hand image, except J and Z, which involve hand motions.

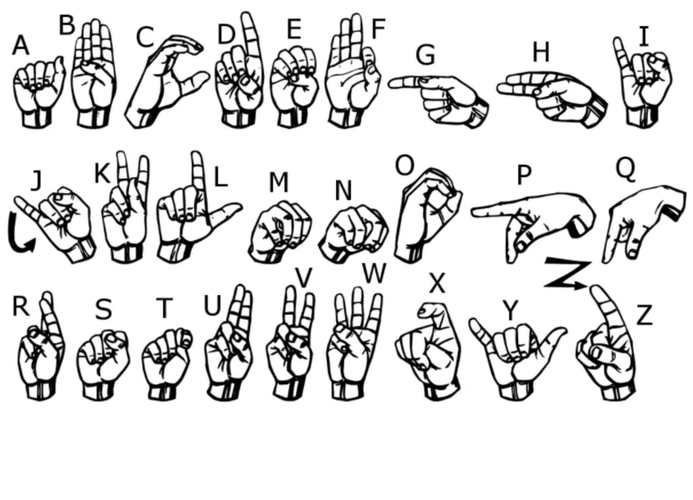

Fig. 1: Source: https://medium.com/@abhkmr30/sign-language-mnist-problem-american-sign-language-48896ea960e0

Our goal

- It is a relevant issue to automate sign language identification, and implement it across several areas. In that way, an image/video of someone displaying a sign, when passed to a machine, can be easily identified, and hence implemented in different fields including gesture and motion recognition.
- We want to compare several neural network models for Sign Language identification, and see how the models perform in order to correctly identify certain hand gestures.
- More complicated models tend to be computationally expensive, and sometimes less generalizable. In contrast, less complicated models often tend to over-simplify the fit. So, we hope to find an optimal model which provides high accuracy overall, without being too time consuming.
- We focus on two types of Neural Network models here: Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN).

Data Description and Pre-processing

- Data description:
  - We use the MNIST Sign Language Dataset available at https://medium.com/@abhkmr30/sign-language-mnist-problem-american-sign-language-48896ea960e0
  - Training dataset: 27,455 hand-sign grayscale images of resolution 28×28.
  - Test dataset consists of 7172 images of resolution 28×28.
  - The numbers 0-25 denote the labels corresponding to the 26 letters of the alphabet. However, there are no images corresponding to the labels $9 = J$ and $25 = Z$, as they denote hand motions which cannot be captured in a single image.
  - Hence, we have 784 features (pixels) corresponding to each of the individual images, which are in one of the 24 classes (excluding J and Z).
- Data Pre-processing:
  - One-hot encoding: Converted the data values to binary vectors for better classification as to bypass the algorithmic prioritization, that is, to treat the response labels as categorical variables.
  - Scaling: Scaled the features (necessary for neural network models) using MinMaxScaler() which kept the values in 0 to 1 range.
- Below, we show some of the images from the training sample.

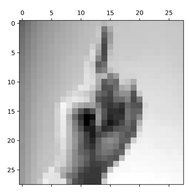

Fig. 2: Sample 1

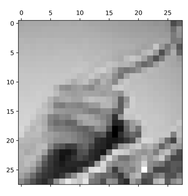

Fig. 3: Sample 2

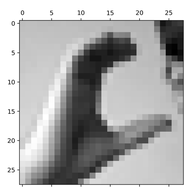

Fig. 4: Sample 3

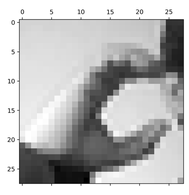

Fig. 5: Sample 4

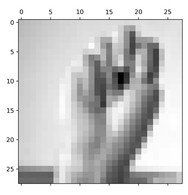

Fig. 6: Sample 5

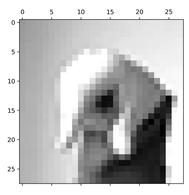

Fig. 7: Sample 6

Models

Artificial Neural Network (ANN)

- We conducted a trial-and-error with the number of hidden layers, the number of neurons in each hidden layer and dropout proportion. The final choices of parameters and hyperparameters are as follows:
- Choice of parameters and hyperparameters in the model:
  - Number of hidden layers: 1 with 350 neurons.
  - Optimizer: Adam and SGD
  - Loss: Categorical Crossentropy
  - Hidden layer activation function: ReLU
  - Output layer activation function: Softmax

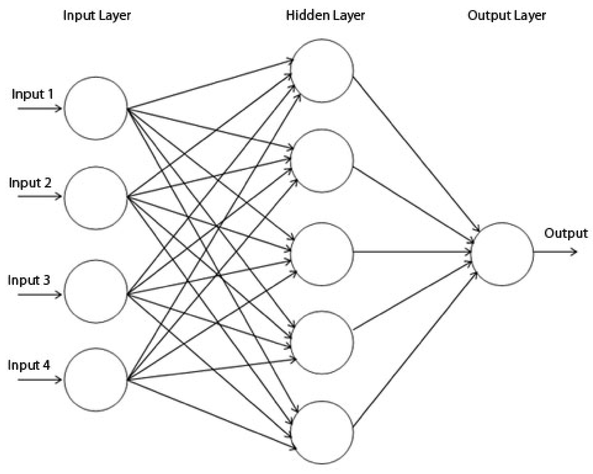

Fig. 8: An Artificial Neural network with a single hidden layer

Convolutional Neural Network (CNN)

- We conducted a trial-and-error with the number of dense hidden layers, the number of neurons in each hidden layer, the number of convolution filters and pooling layers, and dropout proportion. The final choices of parameters are as follows:
- Choice of parameters and hyperparameters in the model:
  - Number of filter layers: 3 of size 3×3
  - Number of filters in each layer: 32
  - Pooling type: Max pooling
  - Dropout: 20 % dropout after second pooling layer
  - Number of hidden dense layers: 1 with 50 neurons.
  - Optimizer: Adam and SGD
  - Loss: Categorical Crossentropy
  - Hidden layer activation function: ReLU
  - Output layer activation function: Softmax
  - Padding: Same

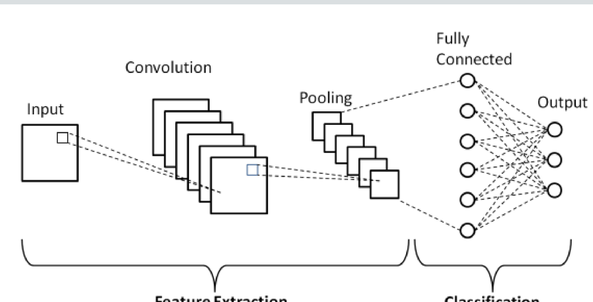

Fig. 9: A Convolutional Neural network

Results

Type of Neural Network	Accuracy
ANN	0.99
CNN	1

Tab. 1: Performance on training data

Type of Neural Network	Accuracy	Weighted Precision	Weighted Recall	Weighted F1
ANN	0.70	0.73	0.70	0.71
CNN	0.92	0.92	0.92	0.92

Tab. 2: Performance on test data

Misclassification Analysis

Accurate by CNN, Inaccurate by ANN

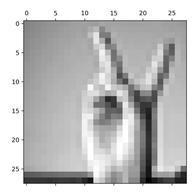

Fig. 10: TRUE: V, ANN: K

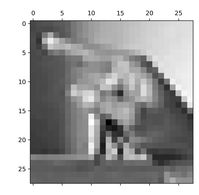

Fig. 11: TRUE: P, ANN: Q

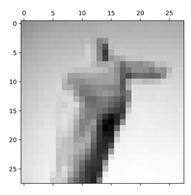

Fig. 12: TRUE: T, ANN: G

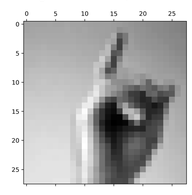

Fig. 13: TRUE: D, ANN: X

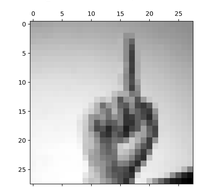

Fig. 14: TRUE: D, ANN: R

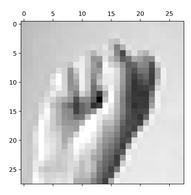

Fig. 15: TRUE: S, ANN: N

Inaccurate by CNN, Accurate by ANN

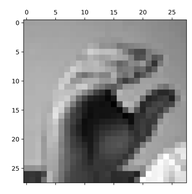

Fig. 16: TRUE: C, CNN: O

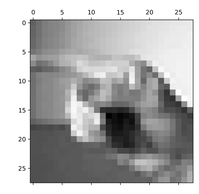

Fig. 17: TRUE: P, CNN: Q

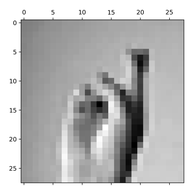

Fig. 18: TRUE: R, CNN: S

Inaccurate by both

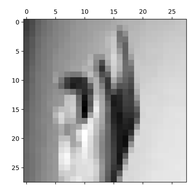

Fig. 19: TRUE: K, ANN: P, CNN: R

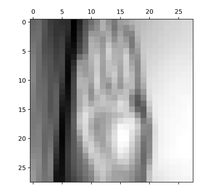

Fig. 20: TRUE: B, ANN: U, CNN: K

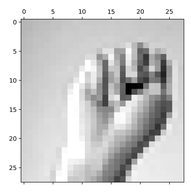

Fig. 21: TRUE: N, ANN: A, CNN: A

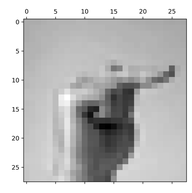

Fig. 22: TRUE: T, ANN: X, CNN: X

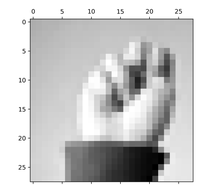

Fig. 23: TRUE: N, ANN: S, CNN: M

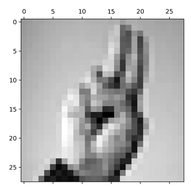

Fig. 24: TRUE: U, ANN: R, CNN: D

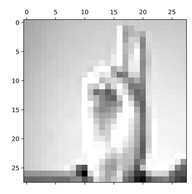

Fig. 25: TRUE: U, ANN: K, CNN: K

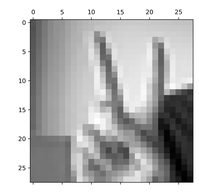

Fig. 26: TRUE: V, ANN: W, CNN: W

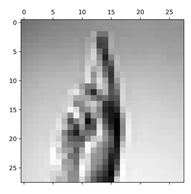

Fig. 27: TRUE: R, ANN: V, CNN: V

Model fit comparison

Accuracy plotted against no. of epochs

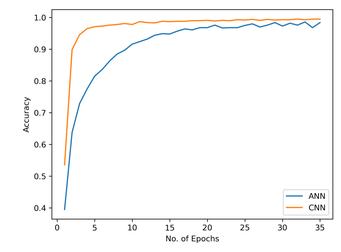

Fig. 28: Optimizer: Adam

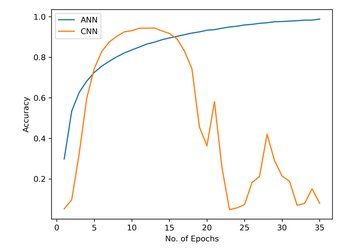

Fig. 29: Optimizer: SGD

Time comparison

Type of Neural Network	Optimizer: Adam	Optimizer: SGD
ANN	387 seconds	449 seconds
CNN	261 seconds	-

Tab. 3: Time (in seconds) to reach 99% accuracy on training data (tried on 35 epochs)

*Optimizer SGD could only attain a maximum of 88% accuracy (after 180 seconds) for CNN.

Type of Neural Network	Optimizer: Adam	Optimizer: SGD
ANN	387 seconds	449 seconds
CNN	40 seconds	113 seconds

Tab. 4: Time (in seconds) taken by CNN to reach 70% accuracy on test data (tried on 35 epochs)

Conclusion

- In this project, we have compared two very popular neural network models ANN and CNN for sign language image identification.
- The results clearly suggest a much better performance by CNN for image classification, compared to ANN, thereby validating the huge popularity of CNN models in image classification and image processing.
- The accuracy for the CNN model is much better compared to the ANN model, and the CNN is also computationally faster.
- SGD Optimizer performed worse compared to Adam for CNN models, however, CNN models using both these optimizers could attain a reasonable testing accuracy in much less time compared to ANN.

Future plan of work

- We want to implement deep learning models which can learn from video data sets and categorize new videos for gesture recognition.
- Gesture recognition has a wide range of applications, and we want to explore data from various areas, or create new datasets to build new deep learning gesture recognition interfaces.

Acknowledgement

I want to send my heartiest gratitude to my sir, Prof. Biswajit Biswas, who suggested me to work on image processing. Secondly, I would like to thank my sister, Indrila Ganguly, for helping me with creating the poster on Overleaf.

References

- Kaggle Dataset: https://www.kaggle.com/datasets/datamunge/sign-language-mnist
- Learn Keras for Deep Neural Networks by Jojo Moolayil.
- Hands-On Data Analysis with NumPy and pandas by Curtis Miller.
- https://medium.com/@abhkmr30/sign-language-mnist-problem-american-sign-language-48896ea960e0