

Machine Learning for ASL Character Classification

1st Adityansh Kapasia
department of Computer Science
University of Florida
Gainesville, USA
adityanshkapasia@ufl.edu

2nd B Pavan Vishnu Sai
department of Computer Science
University of Florida
Gainesville, USA
pavanvishnbetha@ufl.edu

3rd Sahas Gundapaneni
department of Computer Science
University of Florida
Gainesville, USA
sahasgundapaneni@ufl.edu

Abstract—This paper explores the development of a machine learning system designed to classify American Sign Language (ASL) characters. Utilizing a novel dataset comprising images of hand signs representing nine ASL characters, we apply advanced image recognition techniques to accurately identify these characters. Our approach demonstrates a high degree of accuracy in classifying ASL characters, offering the potential for enhanced communication tools for the deaf and hard-of-hearing communities.

Index Terms—deep learning, classification, feature extraction, sign language recognition, depth image, Convolutional Neural Network (CNN), TensorFlow

I. INTRODUCTION

People with hearing and speech disorders use sign language as their primary language. Sign language motions are used by people to communicate their thoughts and emotions nonverbally. Non-signers, on the other hand, find it extremely difficult to comprehend, thus expert sign language interpreters are essential during medical and legal visits, as well as educational and training sessions.

A. Overview of Experiment

1) *Experiment Description*: This study involves developing a Convolutional Neural Network (CNN) model to classify ASL characters. The experiment includes:

- **Data Collection**: Collection of a diverse dataset of images representing nine ASL characters.
- **Model Training**: Training a Sequential CNN model with layers including Conv2D, MaxPooling2D, Dropout, and Dense.
- **Preprocessing Techniques**: Implementation of advanced preprocessing strategies such as:
 - *Image Resizing*: Images are reduced to a consistent dimension (e.g., 100x100 pixels), providing uniformity in the CNN model's input size. This is an important step since it standardizes the data and allows the model to interpret pictures more efficiently and effectively.
 - *Normalization*: The pixel values of the photographs are scaled to a range, often between 0 and 1. Normalization reduces the variability in the dataset, which improves the model's convergence speed during training and contributes to higher overall performance.

- *Random Brightness*: The 'random_brightness' function is used to modify the brightness of pictures at random, improving the model's ability to generalize across varied lighting situations.
- *Adjust Contrast*: The 'adjust_contrast' function alters picture contrast, adding to the model's resilience against variations in image quality and contrast levels.

- **Data Augmentation**: Employing techniques like rotation, width, and height shifting, zooming, and horizontal flipping to increase dataset diversity and reduce overfitting.
- **Model Testing and Validation**: Evaluating the model's accuracy and robustness using unseen data.

B. Literature Review

The development of machine learning models for ASL recognition has been extensively explored in recent literature. Key studies in this field include:

1) *Machine Learning Methods for Sign Language Recognition [1]*: I.A. Adeyanju et al. in their paper *Machine learning methods for sign language recognition: A critical review and analysis*, published in *Intelligent Systems with Applications*, 2021, gives a thorough examination of intelligent systems in sign language recognition during the last two decades. The study emphasizes the complexities of robotic sign language identification as well as the significance of intelligent solutions.

2) *Action Detection for Sign Language Using Machine Learning [7]*: In *Action Detection for Sign Language Using Machine Learning*, S. Urs et al. discuss novel techniques for sign language recognition that use machine learning, as presented at the 2023 International Conference on Network, Multimedia, and Information Technology. This research highlights the power of machine learning in action detection for sign language.

3) *Sign Gesture Classification and Recognition Using Machine Learning [3]*: Muhammad Saad Amin and Syed Tahir Hussain Rizvi, in their 2023 paper published in *Cybernetics and Systems, Sign Gesture Classification and Recognition Using Machine Learning*, using machine learning approaches, investigate classification and recognition of sign gestures. This study adds to our understanding of gesture-based communication systems.

4) *American Sign Language Recognition using Deep Learning and Computer Vision* [4]: Kshitij Bantupalli and Ying Xie, in their 2018 paper *American Sign Language Recognition using Deep Learning and Computer Vision*, presented at the IEEE International Conference on Big Data, explore the use of deep learning and computer vision techniques in the identification of American Sign Language. This research focuses on the use of sophisticated computational approaches to improve the accuracy and efficiency of ASL interpretation.

5) *Comparing ANN, SVM, and HMM based Machine Learning Methods for American Sign Language Recognition* [5]: Rabeet Fatmi et al., in their 2019 paper titled *Comparing ANN, SVM, and HMM based Machine Learning Methods for American Sign Language Recognition using Wearable Motion Sensors*, researchers investigated several machine learning algorithms for ASL identification using motion sensors at the IEEE 9th Annual Computing and Communication Workshop and Conference. This study sheds light on the usefulness of various algorithms in reading sign language using wearable technology.

II. IMPLEMENTATION

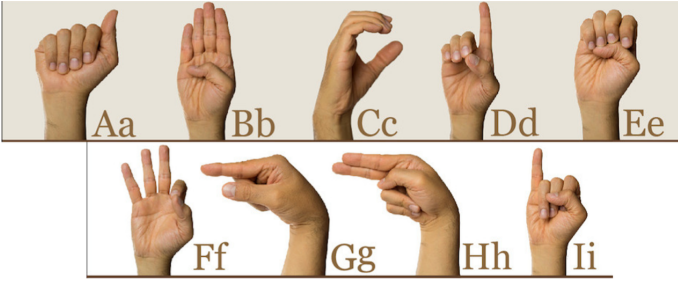


Fig. 1. Examples of ASL character signs.

A. Dataset Description

Data was gathered collectively by group members. Each participant submitted photographs, capturing 10 instances of each of the 9 ASL characters, for a total of 90 images. This amounted to 270 photographs for a group of three.

1) *Character Representation and Encoding*: The dataset focuses on nine distinct ASL characters, each of which is represented by a distinct hand gesture shown in Figure 1. The following table outlines the encoding scheme used for each character:

ASL Character	Integer Encoding
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	7
i	8

TABLE I
INTEGER ENCODING FOR EACH ASL CHARACTER CLASS.

Each image in the dataset captures the palm view of a hand gesture for an ASL character. The images were captured using standard camera-equipped devices such as smartphones. The dataset emphasizes consistency in image resolution and format.

B. Machine Learning Model

For the classification of American Sign Language (ASL) characters, we utilize a custom Convolutional Neural Network (CNN) model. This model is tailored to recognize and classify the hand gestures in the images as specific ASL characters.

1) *Model Architecture*: The architecture of our CNN model comprises a sequential arrangement of layers, each contributing to the extraction and processing of features from the input images. The layers are as follows:

- **Conv2D Layer**: The first layer is a 2D convolution layer with 64 filters of size 3x3, using the 'relu' activation function. It is designed to process the input image of shape 100x100x3.
- **Batch Normalization**: This layer follows each convolution layer to normalize the activations and thus help speed up the training process.
- **MaxPooling2D**: A max pooling layer with a 2x2 window, reducing the spatial dimensions of the input.
- **Additional Conv2D Layers**: Following the first layer, there are additional Conv2D layers with 128, 256, and 512 filters respectively, each followed by batch normalization.
- **Flatten Layer**: This layer flattens the output from the convolutional layers into a 1D array for processing in the dense layers.
- **Dense Layers**: After flattening, the model has two dense layers. The first has 512 neurons, and the second has 256 neurons, both using the 'relu' activation function and L2 regularization to reduce overfitting.
- **Dropout Layer**: A dropout layer with a dropout rate of 0.5 is used between the two dense layers to prevent overfitting.
- **Output Layer**: The final layer is a dense layer with 9 neurons (corresponding to the 9 ASL characters) and uses a 'softmax' activation function for multi-class classification.

Each layer plays a crucial role in the model's ability to learn and dis

III. EXPERIMENTATION

- **Experiment 1: Learning rate optimization** - We started with a high learning rate of 0.0010 with 30 epochs and no callback functions which led to early signs of overfitting such as seeing a very high training set accuracy but bad validation set accuracy and a high validation loss. After reducing the learning rate to 0.0005 we started noticing some improvements and decided to implement call-back functions to adaptively change the learning rate stop the model from training further and stop the epochs at the

best validation accuracy. We utilized two TensorFlow callbacks: EarlyStopping and ReduceLROnPlateau.

- **Experiment 2: Batch size, Epochs** - We experimented with different batch sizes to find the best batch size for our use case and system memory configurations, we found that keeping the batch size of 8 and 16 had very poor accuracy in our dataset, and trying to increase the batch size to more than 32 crashed the kernel. Hence, we decided to keep the batch size at 32. We also experimented with changing the maximum epoch in conjunction with the batch size but having a batch size of 16 or 8 never resulted in a better result even at much higher epochs than having a batch size of 32 and having epochs ranging from 30 to 40. Running the model till epoch 40 showed signs of overfitting and the accuracy plateaued at nearly 90 percent at epoch 35 hence after some fine-tuning of some other hyperparameters as well we decided to have a batch size of 32 and run the model till epoch 35.
- **Experiment 3: Optimizer** - We used the Adam optimizer due to its proven efficiency in handling large and complex datasets [6]. Adam, an algorithm for stochastic optimization is great for its adaptive learning rate capabilities which makes it particularly suitable for the diverse and intricate features encountered in ASL imagery. We did not use Stochastic Gradient Descent (SGD) because it can be slow to converge and sensitive to learning rate settings, Adam ensures faster convergence, which is crucial for efficient model training. Additionally, when compared to other optimizers like RMSprop or Adagrad, Adam demonstrates superior performance in managing sparse gradients, which is a common challenge in image-based classifications.
- **Experiment 4: Dataset loading and Pre-processing**
 - The initial challenge we faced was to load the vast dataset into system memory as the dataset had more than 8000 images of 300 by 300 size, the first experiment was to try and figure out what size of the image should be resized to be used to fit in the system memory without crashing the kernel. The final complex model that we created started to work on an image size of 100 by 100 we tried different sizes such as 200 by 200, 150 by 150, and 110 by 110 but every other crashed the kernel in some interpretation of the model, the best resolution we could find was 100 by 100 that best suited our model. We used the image (library) resize option to resize our images After using the image resize option we faced another issue in normalization where in some cases image resize option would normalize the images already so we added an if else condition to check if the dataset has been normalized or not else we divided by 255 to normalize it.
- **Experiment 5: Convolution Layer Depth Variation** - We assessed the impact of varying depths in the convolutional layers of our model, beginning with a less complex model and incrementally increasing it. We wanted to see how our model performed with different depths of the convolution layers. We found that deeper layers enhanced the model's capacity for complex feature extraction. However, increasing the depth led to an increase in memory consumption. We decided to go with a model architecture comprising 17 layers, including 11 computational layers, balancing between complexity and memory consumption concerning our needs and the size of the dataset.
- **Experiment 6: Activation Function and Regularization Technique** - We used the ReLU activation function as it introduces non-linearity which allows the model to learn complex patterns in sign language images. ReLU also promotes sparsity in the neural network, which leads to more efficient and robust learning making it suitable for complex image recognition tasks like ASL detection [2]. We used L2 regularization in our model's dense layers to examine its impact on overfitting. This technique significantly improved the model's generalization on unseen data, reducing overfitting as shown by better validation performance compared to training data.
- **Experiment 7: Batch Normalization Influence** - We incorporated batch normalization layers after each convolutional layer and prior to the activation functions. The introduction of batch normalization resulted in faster convergence and a more stable training process, indicative of a successful reduction in internal covariate shift within the model.
- **Experiment 8: Flatten Layer Integration** - This layer was crucial for transitioning the output from convolutional layers into dense layers. We observed that flattening significantly increased the number of parameters in the subsequent dense layers, yet it was essential for effectively connecting the convolutional and dense layers within the net.
- **Experiment 9: Dense Layer Configuration** - We varied the number of neurons and layers in the dense part of the network to determine the optimal configuration. Our findings suggested that a configuration comprising fewer but larger dense layers struck an effective balance between model complexity and performance, facilitating efficient learning without overburdening the model.
- **Experiment 10: Dropout Rate Optimization** - We initially set the dropout rate at 0.2 and incrementally adjusted it, observing its impact on the model's performance. Our experiments revealed that a higher dropout rate, up to 0.5, significantly improved the model's generalization capabilities without causing a substantial loss in training accuracy.
- **Experiment 11: Unknown class detection** - In addressing the challenge of classifying an unknown class in American Sign Language (ASL) recognition, we adopt a confidence-based approach, leveraging the intrinsic probability scores of a deep learning model. Post training on known classes (labels 0 to 8), our model, typically a Convolutional neural network, generates predictions accompanied by confidence scores. These scores represent the model's certainty in its classification. To discern

the unknown class (label -1), we set a predefined confidence threshold. During testing, if the model's confidence for a prediction falls below this threshold, we classify the instance as unknown. This method hinges on the assumption that lower confidence scores correlate with instances less similar to the training data, thereby likely belonging to an unlearned, unknown class. We validate this approach by applying it to a mixed dataset of known and unknown classes, fine-tuning the threshold to balance accurately identifying known classes against misclassifying unknowns. This confidence threshold methodology not only enhances the model's robustness but also its applicability in real-world scenarios where encountering unseen data is probable.

IV. CONCLUSION

The project involved the development of a neural network model for American Sign Language (ASL) detection, with its architecture and parameters optimized through a series of experiments. The main emphasis was on improving the accuracy and efficiency of the model by handling overfitting problems, computational limits, and complexities in feature extraction. We began by trying to deal with these problems such as overfitting and model convergence, which we achieved through optimizing the learning rate and integrating callback functions like EarlyStopping and ReduceLROnPlateau. When combined with a learning rate of 0.0005, these callbacks helped to improve the performance of our model on the validation set significantly. Meanwhile, we experimented with batch sizes alongside epochs to settle on 35 epochs at maximum and a batch size of 32 as the optimum balance between accuracy and the system memory limitations. The Adam optimizer was chosen for its superior handling of sparse gradients and adaptive learning rate capabilities, proving more effective than other optimizers like SGD, RMSprop, or Adagrad. In addition, we dealt with issues related to dataset loading and preprocessing where image resizing to 100x100 pixels was found optimal along with conditional normalization for maintaining data quality vis-a-vis computational feasibility. Other refinements were made to the network architecture itself.

REFERENCES

- [1] I.A. Adeyanju, O.O. Bello, and M.A. Adegboye. Machine learning methods for sign language recognition: A critical review and analysis. *Intelligent Systems with Applications*, 12:200056, 2021.
- [2] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019.
- [3] Muhammad Saad Amin and Syed Tahir Hussain Rizvi. Sign gesture classification and recognition using machine learning. *Cybernetics and Systems*, 54(5):604–618, 2023.
- [4] Kshitij Bantupalli and Ying Xie. American sign language recognition using deep learning and computer vision. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4896–4899, 2018.
- [5] Rabeet Fatmi, Sherif Rashad, and Ryan Integlia. Comparing ann, svm, and hmm based machine learning methods for american sign language recognition using wearable motion sensors. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0290–0297, 2019.
- [6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- [7] A S Sushmitha Urs, Vaibhavi B Raj, Pooja S, Prasanna Kumar K, Madhu B R, and Vinod Kumar S. Action detection for sign language using machine learning. In *2023 International Conference on Network, Multimedia and Information Technology (NMITCON)*, pages 1–6, 2023.