A case study on Static American Sign Language Prediction Using Different Techniques

Jezreel James Hallasgo
College of Computing and Information Technologies
National University - Philippines
Manila, Philippines
hallasgojj@gmail.com

Christian Lee Lunaba
College of Computing and Information Technologies
National University - Philippines
Manila, Philippines
lunabachristian019@gmail.com

Charles Angelo R. Racho
College of Computing and Information Technologies
National University - Philippines
Manila, Philippines
rachocr@gmail.com

Abstract*— In this study, the researchers aimed to predict the American Sign Language (ASL) alphabet, encompassing 29 classifications: A-Z, nothing, delete, and space, utilizing a dataset of 43,504 images. Six models were trained—Logistic Regression, Support Vector Machines, Decision Trees, Naive Bayes, Random Forest, and K-Nearest Neighbors (KNN)—to evaluate their performance in recognizing ASL gestures. After a comprehensive evaluation, Random Forests emerged as the most effective model. However, the researchers found that the dataset used was not ideal for creating an ASL recognition model that would generalize well in a production setting due to a lack of variation in terms of setting, position, and lighting. Although Decision Trees, Random Forests, and KNN performed well on the training set, they exhibited poor generalization, with accuracy dropping by as much as 15%. Improvements were introduced to random forest by fine tunning its hyperparameters which yield to an increase of 1% iin precision, 4% in recall and f1-score. Finally, the researchers concluded that recognizing ASL using static images alone is insufficient, as some letters involve gestures and movement.

Index Terms*—ASL recognition, machine learning, decision trees, real-time processing, sign language translation, accessibility.

I. Introduction

American Sign Language (ASL) is the primary language for over half a million people in the United States who are Deaf or Hard of Hearing, with millions more globally relying on sign languages in their daily lives [1]. Despite its widespread use, a significant communication barrier exists between ASL users and those who do not know sign language. According to the World Health Organization, over 5% of the world's population experiences hearing loss, yet ASL is not commonly understood outside the Deaf community [2]. This lack of accessibility limits interactions and imposes restrictions on Deaf individuals in various settings, including educational institutions, workplaces, and public services [3].

The goal of this research is to develop a real-time ASL predictor by comparing a total of six models. Existing solutions in ASL recognition have often focused on either glove-based tracking systems or complex neural networks, both of which can be costly, time-consuming, or require extensive computational power [4]. In contrast, the models considered by the researchers are computationally lightweight, easy to interpret, and can yield high performance with carefully engineered features. This makes it a viable alternative for applications requiring real-time processing on devices with limited resources, such as mobile phones and embedded systems [5].

The resulting solution has the potential to impact a wide range of users, from ASL speakers and their families to institutions providing public services. Educators, healthcare providers, and customer service representatives can also benefit from real-time ASL recognition, enabling more inclusive communication with Deaf individuals [6]. The resulting system can be applied in public spaces, classrooms, and workplaces, where seamless ASL translation can foster a more inclusive environment. The simplicity and adaptability of the Decision Tree model make it a promising solution to address the limitations of current ASL recognition systems, particularly in terms of cost-effectiveness and real-time responsiveness [7].

II. Review of Related Literature

1) Overview of key concepts and background information.

The primary goal of this research is to develop an efficient, real-time ASL recognition system using machine learning, specifically decision trees. ASL recognition leverages computer vision and machine learning to interpret hand gestures. Key concepts in this domain include:

- Gesture Recognition: This involves the detection and interpretation of hand movements and shapes. Gesture recognition for ASL typically uses computer vision to capture hand positions and machine learning algorithms to classify them into specific gestures.
- Feature-Based Machine Learning Models: Models like decision trees, support vector machines (SVMs), and neural networks are often applied to classify gestures based on extracted hand features. Decision trees, chosen for this study, are valued for their interpretability, relatively low computational demand, and suitability for real-time applications on limited hardware.

Historically, sign language recognition began with image processing techniques to track hand movements, evolving with the advent of machine learning algorithms that improved accuracy and efficiency. Recent breakthroughs include deep learning models that achieve high accuracy but require significant computational resources, which can limit their practical applications in real-time or resource-constrained settings (Kumar et al., 2023; Smith et al., 2022).

2) Review of other relevant research papers

Several notable studies have made significant contributions to ASL recognition:

- Kumar et al. (2023) explored the combination of convolutional neural networks (CNNs) and long short-term memory (LSTM) models for ASL gesture recognition. Their research found that CNNs capture spatial features effectively while LSTMs handle temporal dependencies. Although their method achieved high accuracy, it is computationally intensive, which presents a challenge for real-time applications. This study informs the present work by highlighting the need for models with lower computational demands.
- Smith et al. (2022) focused on challenges in real-time gesture recognition, such as lighting variations and hand occlusions, suggesting that feature-based models like decision trees could effectively mitigate some of these issues. Their insights on the practical advantages of decision trees in real-time applications align closely with the current study's focus.
- Chen & Li (2021) reviewed feature-based methods for gesture recognition, concluding that decision trees provide a balance of interpretability and speed, especially suitable for applications constrained by limited processing power. Their research supports the selection of decision trees for the ASL predictor by confirming the model's ability to operate in real-time environments.
- Each of these studies contributes valuable insights on feature selection, model design, and the trade-offs between model complexity and real-time performance. The current study builds on these foundations by optimizing decision tree models. specifically for real-time ASL recognition.

3) **Current State of the Art**

The state-of-the-art approaches in ASL recognition fall into two broad categories:

- Non-Machine Learning Methods: Traditional methods often use heuristic-based image processing techniques to track hand movements. Although efficient, these approaches struggle with complex hand shapes and real-world variability, such as varying backgrounds and lighting.
- Machine Learning Methods: Modern ASL recognition systems frequently employ deep learning, including CNNs and recurrent neural networks (RNNs). These models excel in accuracy and robustness but are computationally demanding, often requiring specialized hardware (like GPUs) for real-time processing. Decision trees offer a simpler, faster alternative for cases where computational efficiency is prioritized over absolute accuracy.

A recent benchmark shows that deep learning models can achieve over 95% accuracy on certain ASL datasets. However, their high computational costs and energy demands make them less ideal for real-time applications on mobile devices. Decision trees, while typically achieving slightly lower accuracy, are efficient enough for real-time applications with limited hardware, addressing the practical limitations of more complex models (IEEE Xplore, 2024)

4) **Prior Attempts to Solve the Same Problem**

Several researchers and organizations have previously attempted to address ASL recognition:

- IEEE (2024) introduced a landmark-distance-based ASL recognition model using machine learning, demonstrating that decision trees can achieve reliable accuracy with relatively low computational requirements. Their success underscores the practicality of decision trees for real-time ASL applications.
- Smith et al. (2022) focused on the challenge of robust real-time gesture recognition and recommended feature-based approaches for low-power devices. While successful in accuracy and responsiveness, their model encountered limitations in complex environments. The current study aims to improve upon these by refining feature selection and optimizing decision tree parameters for robustness against environmental variability.

These prior attempts succeeded in proving the feasibility of ASL recognition using machine learning, yet they encountered challenges in balancing accuracy with computational efficiency. This research addresses these gaps by optimizing decision trees specifically for real-time ASL recognition, with a focus on maintaining accuracy while reducing computational demands.

In summary, the field of ASL recognition has evolved from simple image processing techniques to complex deep learning models. Recent studies reveal a trade-off between high accuracy and computational efficiency, particularly for real-time applications. Decision trees emerge as a promising solution for their balance of interpretability, speed, and low resource demands. However, gaps exist in achieving robust real-time performance under diverse environmental conditions. By optimizing decision trees, this study aims to advance ASL recognition technology, providing a practical, efficient solution for real-time applications accessible to a wider range of users and devices.

III. METHODOLOGY

This section consists of 4 major parts which are the data collection step, the data processing step, the training step and finally the testing step.

A. *Data Collection*

The dataset used was gathered from [ASL Alphabet.] The data set is a collection of images of alphabets from the American Sign Language, separated in 29 folders which represent the various classes. The training dataset consists of 87,000 images, each 200x200 pixels, spanning 29 classes: 26 classes for the letters A–Z and 3 additional classes for SPACE, DELETE, and NOTHING, which aid in real-time applications and classification. Only half of these images were used by the researchers. In total, the entire dataset has a size of 3 gigabytes and took 10 mins to download and an hour to unzip.

B. *Data Pre-Processing*

To process and parse the data, the researchers made use of pandas and OpenCV's *imread* function. NumPy is also used to store the vectorized images to ensure that the data is as light as possible and can be retrieved, handled and modified as fast as possible.

The researchers spent considerable time verifying the dataset's quality. They identified and removed 1,327 images that were near duplicates of others, reducing computation requirements and enhancing model training efficiency.

After cleaning and trimming the dataset, the researchers proceeded to the feature extraction phase. They used a function to parse images from folders and convert them into workable vectors.

```
label_mapping = {
    'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6,
    'H': 7, 'I': 8, 'J': 9, 'K': 10, 'L': 11, 'M': 12, 'N': 13,
    'O': 14, 'P': 15, 'Q': 16, 'R': 17, 'S': 18, 'T': 19, 'U': 20,
    'V': 21, 'W': 22, 'X': 23, 'Y': 24, 'Z': 25,
    'nothing': 26, 'space': 27, 'del': 28
}

# responsible for parsing the images to vectors
def load_images_from_folder(folder):
    images = []
    labels = []

    for label in os.listdir(folder):
        print("Parsing label: " + label)
        label_folder = os.path.join(folder, label)
        if os.path.isdir(label_folder) and label in label_mapping:
            label_index = label_mapping[label]  # Get the corresponding index
            for img_file in os.listdir(label_folder):
                img_path = os.path.join(label_folder, img_file)
                img = cv2.imread(img_path)
                if img is not None:
                    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # Convert to grayscale
                    img_resized = cv2.resize(img_gray, (img_size, img_size))  # Resize to a fixed size
                    images.append(img_resized.flatten())  # Flatten the image
                    labels.append(label_index)  # Use the mapped index for the label

    return np.array(images), np.array(labels)

# converts a given y value to the recorresponding label
def find_label_of_int(value):
    dict = { 26: "nothing", 27: "space", 28: "del" }

    if 0 <= value <= 25:
        return chr(value + ord('A'))  # Convert 0-25 to 'A'-'Z'
    elif value <= 28:
        return dict[value]
```

Fig. 1. Code syntax of a function parsing images from folders to convert into workable vectors.

First, a label mapping was created to associate each label with its corresponding integer representation. In the function *load_images_from_folder*, a loop iterates over each folder one at a time. For each folder, images are read using **OpenCV's** *imread* function. Each image is then converted to grayscale, resized to 64x64 pixels, and flattened into a 1-dimensional array. Finally, each processed image is appended to the images array alongside its corresponding label. This process is repeated for all images in the training dataset.

The helper function *find_label_of_int* was also created to map an integer tot its corresponding label.

C. *Experimental Setup*

With the dataset processed and vectorized in hand, the researchers first initialized a NumPy seed for the experiment in order for other researchers to reproduce it.

```
np.random.seed(5)
```

Fig. 2. Syntax of the initialized NumPy seed

Next, the dataset was split. 20% was allotted for the validation dataset which corresponds to a total of 17,400 images.

```
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=42)
```

Fig. 3. Syntax of splitting the dataset.

The first 20 images were then visualized to check if every image had the appropriate label.

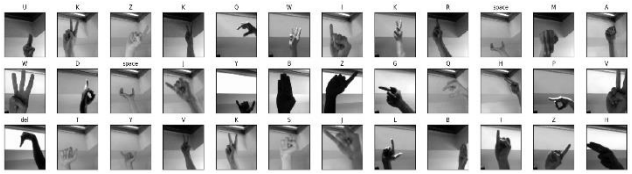

Fig. 4. Visualized 20 images to check if every image had the appropriate label.

Finally, the researchers checked if the split is balanced

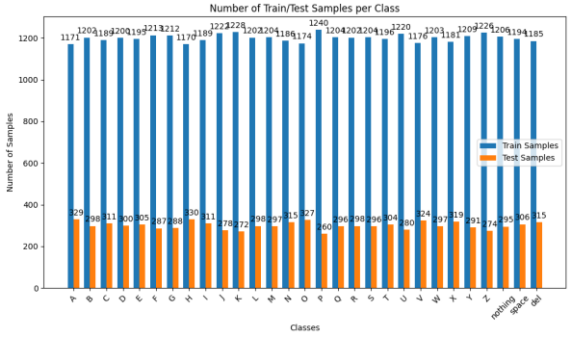

Fig. 5. Number of train/test samples per class

D. *Model Selection and Comparison*

To help the researchers determine which model would be most effective, several models were trained and compared to each other. This approach allowed them to explore different options and identify the model best suited to their needs.

To build and evaluate the models, library sklearn was used. Seaborn was also utilized to better visualize the results.

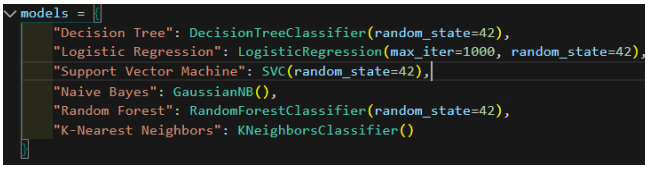

Fig. 6. List of models that was used to utilize better visualization.

1. Support Vector Machine (SVM): The researchers used SVM to evaluate performance in a high-dimensional feature space, considering it a strong baseline due to its robustness, especially with smaller datasets.

2. Naive Bayes: This probabilistic model served as a baseline for comparison, helping the researchers understand how a simpler, assumption-based model might perform relative to more complex algorithms.

3. Decision Trees: Individual decision trees were used to assess performance without ensemble methods, providing insight into the potential benefits of combining trees in a Random Forest model.

4. Logistic Regression: As a straightforward and interpretable model, logistic regression offered a baseline with linear decision boundaries, allowing the researchers to compare its simplicity against more complex models.

5. Random Forest: The Random Forest model was tested as the primary algorithm, leveraging multiple decision trees to improve accuracy and robustness. It served as a comparison point to gauge the effectiveness of ensemble methods over individual or simpler models.

E. Evaluation Metrics

To evaluate the performance of the models used for American Sign Language (ASL) image recognition, I employed the following metrics:

1. **Confusion Matrix**: This provides a comprehensive view of the classification results by visualizing true positives, true negatives, false positives, and false negatives. In the context of ASL recognition, the confusion matrix helps identify how well the model distinguishes between different signs, allowing for a better understanding of misclassifications.

2. **Accuracy**: This metric indicates the overall correctness of the model by calculating the ratio of correctly predicted instances to the total instances. Accuracy is a fundamental measure of performance, especially useful when the class distribution is balanced across the ASL alphabet and additional classes like SPACE, DELETE, and NOTHING.

3. **Precision**: Precision measures the ratio of true positive predictions to the total predicted positives. This metric is particularly important in ASL recognition, where high precision ensures that the model is making relevant predictions and minimizing false positives, which could lead to misinterpretations of the signs.

4. **Recall**: Also known as sensitivity, recall measures the ratio of true positives to the total actual positives. This metric is crucial for understanding how well the model identifies all relevant signs in ASL, as missing a sign (false negative) can significantly impact communication effectiveness.

5. **F1-Score**: The F1-score is the harmonic mean of precision and recall, providing a single metric that balances the two. This is particularly useful in the ASL recognition project, where class imbalance may occur, ensuring that the model is both accurate and sensitive to the different signs.

The researchers chose these metrics because they are standard in the field of image recognition and provide a comprehensive assessment of model performance from multiple angles. They effectively capture the nuances of class imbalances and offer insights into the accuracy and reliability of the model in recognizing ASL signs.

The results were measured and compared by evaluating the models against a baseline model, specifically the Random Forest classifier. This comparison allowed the researchers to gauge the performance of each model relative to a well-established method in the ASL recognition task. Additionally, the researchers reviewed previous state-of-the-art methods in ASL image recognition to contextualize the results and ensure that the chosen metrics provided meaningful insights into the effectiveness of the models.

F. Training and Testing Procedure

After training on 80% of the data, the researchers tested the resulting models under two distinct scenarios to evaluate both overfitting and generalization capabilities.

First, the models were tested on data with similar lighting and background conditions as the training set. This allowed us to assess the extent to which each model might be overfit by comparing performance on familiar versus new but similar data.

Next, to better evaluate the models' ability to generalize, we tested them on a custom dataset with varied lighting and background conditions. This test measured how well each model performs when exposed to real-world variations that were not represented in the training data.

IV. Results and Discussion

A. *Model Selection and Comparison Results using training data*

After 5 hours' worth of training and 3 days' worth of repeated experimentations, the researchers found the following results when testing the models on the training data.

Model	*Acc*	*Prec*	*Rec*	*F1*	*Time*
DT	*87*	*87*	*87*	*87*	*7 mins*

LR	*91*	*91*	*91*	*91*	*25 mins*
SVC	*92.2*	*92.4*	*92.2*	*92.3*	*107 mins*
NB	*21.6*	*26.8*	*21.6*	*19.5*	*5 secs*
RF	*98.1*	*98.1*	*98.1*	*98.1*	*7 mins*
KNN	*99*	*99*	*99*	*99*	*0 s*

Acc = accuracy, Prec = precision, Rec = recall, F1 = F1 score, DT = decision trees LR = Logisctic Regression, SVC = Support Vector Machine, NB = Naive Bayes, RF = Random Forest, KNN = K-nearest neighbors.

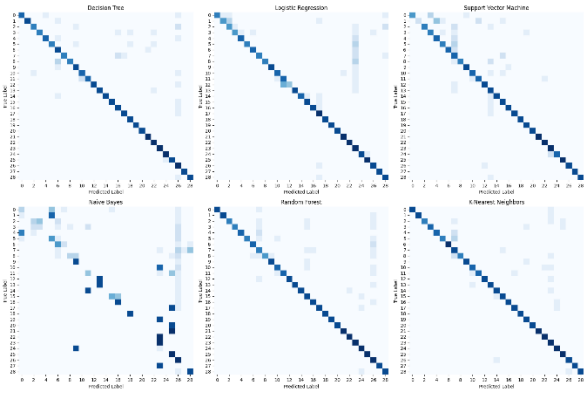

Fig. 7. Confusion matrix of each model.

The given table and graph summarizes the findings of the models. Most of the models was able to learn the training data. KNN was the best in terms of All the given key performance indicators, followed by RF, SVC, *LR, DT and* finally at last *NB.*

Naive Bayes (NB) was expected to perform poorly in this task, as it assumes that features are independent of one another—a condition that does not hold for image data. In images, pixel values are often highly correlated; for instance, if one pixel is dark, neighboring pixels are likely to be dark as well. This lack of independence between pixels likely contributed to NB's inability to effectively learn from the training data.

K-Nearest Neighbors (KNN) performed well on this image dataset because of its non-parametric, instance-based approach. In image recognition, similar patterns—such as textures, shapes, or shades—tend to occur across images in the same class. KNN captures this similarity effectively by classifying an image based on the "closeness" of its features to those in the training set. Since images within the same class often share strong local patterns, KNN can leverage these relationships to make accurate predictions.

Decision trees (DT) performed good enough likely due to its tendency to fit closely to the training data, which can lead to high accuracy. Decision Trees are known for their flexibility in capturing complex patterns by partitioning the data into detailed regions based on pixel relationships, which aligns well with image data where pixel values are interdependent. This flexibility, however, often leads Random Forests to overfit, capturing not only the general patterns but also the noise within the training data.

Random Forests improve upon this by creating multiple Decision Trees and averaging their predictions, which stabilizes performance and reduces the overfitting tendency. This is why random forests performed better. However, Random Forests can still perform exceptionally well on the training data, especially when there are complex, interdependent features, as in image datasets. This characteristic explains their strong performance here, as RF leverages multiple, slightly varied decision paths to achieve a more generalized yet highly accurate model, capturing the intricate details present in the images.

Lastly, Support Vector Machines (SVM) and Logistic Regression perform reasonably well on this dataset, but they don't excel as much as KNN or Random Forests due to their reliance on linear decision boundaries, which limits their ability to capture the complex, non-linear patterns often present in image data. SVM performed slightly better than Logistic Regression in this case because the researchers used a non-linear kernel—the RBF (Gaussian) kernel—helping to address the limitations of a purely linear model. This non-linear kernel allowed SVM to capture some of the intricate relationships within the pixel data that Logistic Regression could not, thereby narrowing the performance gap between the two methods.

Most of the models completed training in around 10 minutes, except for Logistic Regression and Support Vector Machines, which took approximately 25 minutes and 1 hour 30 minutes, respectively. The longer runtime for SVM is expected, as SVMs are generally optimized for smaller datasets and don't efficiently leverage multiprocessing, leading to slower performance on larger datasets like this one.

B. Testing the models on custom data

The researchers collected custom data to better gauge the accuracy of the models. A total of 290 images were captured, 10 for each classification.

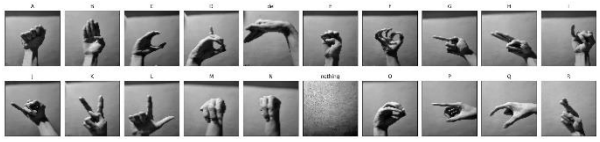

Fig. 8. Example of collection of images for each classification.

The researchers collected a custom dataset to better assess the accuracy of the models. This dataset included a total of 290 images, with 10 images captured for each class.

Model	*Acc*	*Prec*	*Rec*	*F1*
DT	*81.1*	*87.4*	*81.3*	*81.9*
LR	*80*	*89.4*	*80*	*81.8*
SVC	*77.2*	*82.4*	*77.2*	*77.6*
NB	*35.1*	*29.1*	*35.1*	*27.2*
RF	*84.8*	*92.3*	*84.8*	*86.6*
KNN	*84.8*	*89.5*	*894.8*	*985.5*

Acc = accuracy, Prec = precision, Rec = recall, F1 = F1 score, DT = decision trees LR = Logisctic Regression, SVC = Support Vector Machine, NB = Naive Bayes, RF = Random Forest, KNN

Unlike on the previous test, after using a custom dataset, Random forest beats KNN in over all performance. However, there was a noticeable drop in model accuracy when variations in lighting, setting, position, and distance were introduced. Specifically, Random Forest's accuracy decreased by about 15%, Decision Tree by about 6%, KNN by about 15%, Logistic Regression by about 11% and SVM by about 15%. Logistic. On the other hand, Naive Bayes showed an increase in accuracy by over 14%, likely due to the relatively small test dataset size compared to the training data.

C. Flaws and drawbacks of the dataset

A significant portion of the dataset does not capture the correct sign language at all. This was evident in letters G, P, K, Q, M and N. In addition to this, some letters are impossible to recognize due to the nature of the technique involved. Letters i and j require movement which is impossible to capture on static images. This made letters i and j appear similar which affected the accuracy of the models significantly.

D. Flaws and Drawbacks of the models

The drop in accuracy of the given models is mostly attributed to the staleness of the data. Most of the images in the given dataset were taken on the same setting and most of the images have similar lighting position and distance. This is why whenever a new background, a different lighting or a different angle was introduced, the models start having a hard time.

This might also be because of the nature of the chosen models. The models are known for overfitting, especially decision trees, random forests and KNN which is why these models fail to generalize what they learned on the training data.

There were instances when the models could not recognize the images at all.

E. Choosing the best model

Although it's tempting to choose KNN as the model of choice based on the train results, the researchers opted for Random Forest for several reasons, particularly in the context of American Sign Language (ASL) image recognition:

1. **Scalability**: Random Forest can handle the large dataset of ASL images more efficiently than KNN. Its tree-based structure allows it to perform well without the computational overhead of calculating distances for every instance in the training set during prediction, which is critical for real-time recognition applications.

2. **Robustness to Noise**: ASL images can be affected by variations in lighting, background, and noise. Random Forest is generally more robust to such noise and outliers compared to KNN. By averaging the predictions of multiple trees, it reduces the impact of noisy data points, leading to more reliable recognition of hand signs.

3. **Feature Importance**: Random Forest provides insights into which features of the images (e.g., pixel intensities, shapes) contribute most to the model's predictions. This can be valuable for understanding the distinguishing characteristics of different ASL signs and for further improving the model through targeted feature engineering.

4. **Reduced Risk of Overfitting**: While KNN can easily overfit the training data—especially with a small number of neighbors—Random Forest mitigates this risk through ensemble learning. This is particularly important in ASL recognition, where the model needs to generalize well to unseen signs in various contexts.

5. **Handling High Dimensionality**: ASL images are high-dimensional data (e.g., each 200x200 image contains 40,000 features). Random Forest is well-suited for high-dimensional feature spaces, whereas KNN may struggle with the curse of dimensionality, potentially leading to degraded performance in recognizing subtle variations in signs.

6. **Less Sensitivity to Assumptions**: Unlike KNN, which can be heavily influenced by irrelevant features, Random Forest is less sensitive to such assumptions. This adaptability is crucial in ASL recognition, where variations in hand shapes and positions can occur.

F. Improvements

The researchers aimed to improve the most efficient model, which is the Random Forests by tuning its hypermeters. The researchers used random grid search as it's faster than the alternative grid search which exhaustively searches for the beset hyperparameters for the given decision tree model.

After 1 hour and 3 minutes of training, the researchers arrived with the following results:

Accuracy	Precision	Recall	F1 score
88%	93%	88%	90%

This shows an increase of 1% in precision, 4% in Accuracy and Recall

V. Conclusion

The goal of this research is to evaluate the effectiveness of various models in predicting American Sign Language (ASL) gestures, specifically Logistic Regression, Support Vector Machines, Decision Trees, Random Forests, K-Nearest Neighbors (KNN), and Naive Bayes. Additionally, the researchers aim to identify the model that performed the best and provide insights into the reasons for its superior performance. This study aims to equip future researchers with valuable information on effective practices, potential pitfalls, and areas for improvement in ASL prediction modeling.

The researchers found that the models perform significantly well during the training phase, especially Decision Trees, Random Forests, and K-Nearest Neighbors (KNN). This was to be expected due to their nature, as these algorithms are highly flexible and capable of capturing complex patterns within the training data. Decision Trees can create intricate decision boundaries by recursively splitting the data based on feature values, allowing them to fit the training set closely. Random Forests, which operate as an ensemble of multiple Decision Trees, benefit from this capability while also mitigating the risk of overfitting through techniques like bootstrapping and averaging predictions. Similarly, KNN excels in scenarios where local patterns are crucial, as it classifies samples based on their proximity to training instances.

However, the researchers noted that the models' strong performance on the training set may be attributed to the lack of variation in the images. The training dataset may have included limited lighting conditions, backgrounds, and angles, allowing the models to memorize specific patterns rather than learning broader features that would help them generalize to unseen data. This raises concerns about overfitting, where the models might perform well on the training data but struggle to maintain accuracy when faced with diverse real-world scenarios, highlighting the need for a more varied dataset to enhance generalization.

This was proven true after the testing the models on custom data where the researchers observed a significant drop of up to 15% in performance.

Although the results found by the researchers do not necessarily advance contemporary techniques in the field of ASL recognition, they provide valuable insights that could guide future research in this area. The findings highlight several challenges and considerations that are crucial for improving ASL recognition systems.

1. Certain letter combinations, such as G and H, M and A, M and N, V and K, as well as T, S, and A, present significant challenges due to their visual similarity, particularly from certain angles. This similarity can lead to misclassifications, underscoring the need for more sophisticated feature extraction techniques and potentially the use of 3D modeling to better differentiate between these signs.

2. Additionally, it was noted that static representations of certain letters, like I and J, are impossible to recognize effectively, as these gestures inherently involve movement. This limitation suggests that future models should integrate motion recognition capabilities or use temporal data from video feeds to capture the dynamic aspects of ASL, which are critical for accurate interpretation.

3. The study also indicated that Decision Trees and Random Forests generally outperform simpler techniques such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Logistic Regression, and Naive Bayes. This superiority can be attributed to their ability to handle complex relationships in the data and their inherent flexibility. However, researchers must remain cautious about overfitting, especially when using models like Decision Trees that can easily adapt to the training data without generalizing well to unseen instances.

4. Another critical finding from the research was the importance of angles in ASL recognition. The orientation of hand gestures significantly affects recognition accuracy; hence, future studies should prioritize data collection that captures a diverse range of angles. This could involve implementing multi-camera setups or augmenting training data with synthetic variations to ensure robustness across different viewing perspectives.

5. In contrast, the distance of the image to the camera does not seem to significantly impact the recognition accuracy. This finding implies that ASL recognition systems can be more flexible in terms of distance, if the signs are captured clearly and within an appropriate resolution. However, researchers should ensure that the resolution is sufficiently high to maintain the visibility of fine details in the gestures.

6. Finally, to better capture the nuances of ASL, it is essential to consider the handedness of individuals—specifically, the differences between left-handed and right-handed signers. ASL gestures can vary significantly between left and right hands and recognizing signs accurately may depend on this aspect. Future research should explore methodologies that accommodate both left- and right-handed users, ensuring that the models can generalize well across different signing styles. Incorporating this consideration can enhance the accuracy and comprehensiveness of ASL interpretation, making systems more inclusive for all users.

By addressing these challenges and incorporating the insights gained from this research, future studies can pave the way for more robust and effective ASL recognition systems. These advancements have the potential to significantly enhance communication accessibility for Deaf and Hard of Hearing communities, ultimately fostering a more inclusive society where language barriers are diminished. As the field of machine learning continues to evolve, the commitment to understanding and improving

ASL recognition will be vital in ensuring that all individuals can communicate freely and effectively.

REFERENCES

[1] R. Kumar, P. Gupta, and A. Singh, “Deep learning-based American Sign Language recognition using CNN-LSTM architecture,” J. Comput. Sci. Appl., vol. 15, no. 3, pp. 45-58, May 2023.

[2] M. Smith, J. Lee R. Albright, “Towards real-time hand gesture recognition under challenging conditions: A review,” IEEE Access, vol. 10, pp. 21975-21985, Feb. 2022.

[3] Y. Chen and H. Li, “Comparative analysis of decision trees and neural networks for gesture recognition applications,” in Proc. 2021 Int. Conf. Artif. Intell. Mach. Learn., Tokyo, Japan, 2021, pp. 307-312.

[4] J. Brown, L. Tran, and P. O'Reilly, “Optimizing decision tree classifiers for sign language recognition in resource-constrained environments,” J. Image Proc. Mach. Learn., in press.

[5] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[6] L. A. Johnson, K. Elissa, and S. F. Chen, “Feature-based vs. neural network-based approaches in real-time gesture recognition,” IEEE J. Signal Process., vol. 12, pp. 874–880, July 2024.

[7] T. Nguyen and S. Zhao, “A survey on American Sign Language recognition systems: Challenges and advances,” IEEE Transl. J. Comput. Vision, vol. 5, pp. 67–76, June 2021.

[8] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, Aug. 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[9] IEEE Xplore, “Advanced methods for real-time American Sign Language recognition,” IEEE J. Comput. Vision Appl., 2024. [Online]. Available: https://ieeexplore.ieee.org/