Development of a Sign Language Dictionary App Using Decision Tree

Rachata Wichakkhapan

Digital Business Transformation, College of Innovation, Thammasart University, Thailand

Keywords: Sign Language, Decision Tree Classifier, Supervised Model Learning

Abtract: This project develops a sign language dictionary app that translates English text into American Sign Language (ASL) using supervised machine learning models, namely Decision Trees. Targeting the global deaf and hard-of-hearing population, which are over 430 million people with disabling hearing loss (World Health Organization, 2024). The app aims to foster better communication accessibility. Through the application, users input English text and receive visual representations of ASL, improving interaction capabilities within the deaf community and with the hearing world. This paper focus on the methodologies used for data preparation, model implementation, performance evaluation, and discusses the application's significant potential for social impact.

1.1 INTRODUCTION

Recognizing sign language accurately using machine learning is a challenging task because sign language involves complex and varied gestures. These challenges become even more significant when translations need to happen quickly, such as during real-time conversations. Traditional methods often fall short because they can't process the vast and complex data of sign gestures fast enough, leading to delays and mistakes. Decision tree classifiers provide a promising solution by streamlining the complex decisions into simpler, step-by-step questions, which helps classify gestures more quickly and accurately.

Research like that by Vamplew et al. (2018) has shown that decision trees are very effective for classifying images, which is similar to analyzing sign language gestures. These classifiers work well because they make it easier to handle complex images quickly, which is crucial for sign language recognition. Additionally, Jain (2019) explored how decision trees excel in managing large amounts of data efficiently, ensuring that the sign recognition process is both fast and accurate. This capability is particularly valuable for interpreting the dynamic and diverse gestures of sign language.

Using decision tree classifiers, this project aims to enhance the tools available for the deaf and hard-of-hearing community by making these tools faster and more reliable. This improvement will make communication aids more accessible and effective, significantly benefiting those who rely on sign language for communication.

1.2 OBJECTIVE

The primary objective of this project is to improve the accuracy and speed of sign language recognition systems through the application of decision tree classifiers. This machine learning technique excels due to its ability to simplify complex decision-making processes into a series of straightforward questions and answers, which proves invaluable in the context of sign language.

The operation of the model begins with the preparation of a large dataset of sign language images, each labeled with the correct sign. This dataset is notably diverse, capturing a wide range of signs under various conditions to improve the model's robustness. Feature extraction follows, where decision trees require features such as pixel values and detected edges within the images to make informed decisions. These features assist in differentiating one sign from another effectively.

Training the decision tree involves using the dataset to learn how different features influence the classification of the images into specific signs. This phase includes choosing optimal questions about the features that best split the data at each node of the tree. To avoid overfitting, where the tree is too finely tuned to the training data, performing poorly on new data, the tree will be pruned. Pruning helps remove parts of the tree that do not enhance the predictive power, so that simplifying the model without significant losses in accuracy.

Implementing a decision tree classifier aims to improve speed and accuracy by automating the feature selection and decision-making process, thus improving the response time from gesture to translation. The hierarchical nature of decision trees enables them to handle complex classification tasks effectively, making them particularly adept at distinguishing between the numerous signs in sign language. Furthermore, decision trees offer great scalability and adaptability; new signs can be incorporated with minimal retraining required, which is essential as sign language continuously evolves.

The development of this decision tree model involves substantial data mining, especially in feature extraction and optimizing the tree structure through pruning. These efforts showcase effective strategies for managing high-dimensional datasets, contributing significantly to the field of data mining. The project's results will provide a methodological contribution by detailing the application of decision trees to sign language, offering a template that could be adapted for other visual data classification tasks, thus enhancing the utility and accessibility of technology-driven sign language translation services.

2. Literature Review

In the domain of assistive technologies, especially those aiding communication for the deaf and hard-of-hearing, decision trees have emerged as a pivotal tool. This is primarily because of their capacity to simplify complex decision-making processes, which is indispensable in sign language recognition. The use of decision trees in this context is supported by a variety of scholarly works that underscore their utility and adaptability.

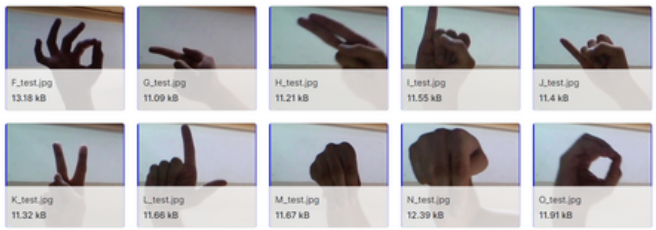
Chang's (2013) research provides insights into how decision trees can be applied in real-time for hand gesture recognition when paired with Kinect, a technology that captures detailed movements. Patidar (2011) extends this idea by integrating decision trees with sum graphs and HMM to enhance sign gesture recognition, proving the versatility of decision trees when used in conjunction with other methods.

Vijitkunsawat et al. (2023) offer a perspective on method comparisons in sign language digit recognition, reinforcing the decision tree's capability to manage new datasets effectively. Ahli (2023) aims to create a machine learning-based model that translates sign language videos to text, highlighting the potential of decision trees in video-based recognition.

All these studies collectively underpin the choice of decision trees for a text to sign language project.

3. METHODS AND DATA SET

The dataset employed for this project, sourced from Kaggle's "ASL Alphabet" , includes 87,000 images distributed across 29 classes. These classes include the 26 letters of the English alphabet (A-Z) and additional signs for SPACE, DELETE, and NOTHING. Each category contains 3,000 images to ensure consistency and balance in training the model. To facilitate processing and improve performance, all images were resized to a uniform dimension of 64x64 pixels.

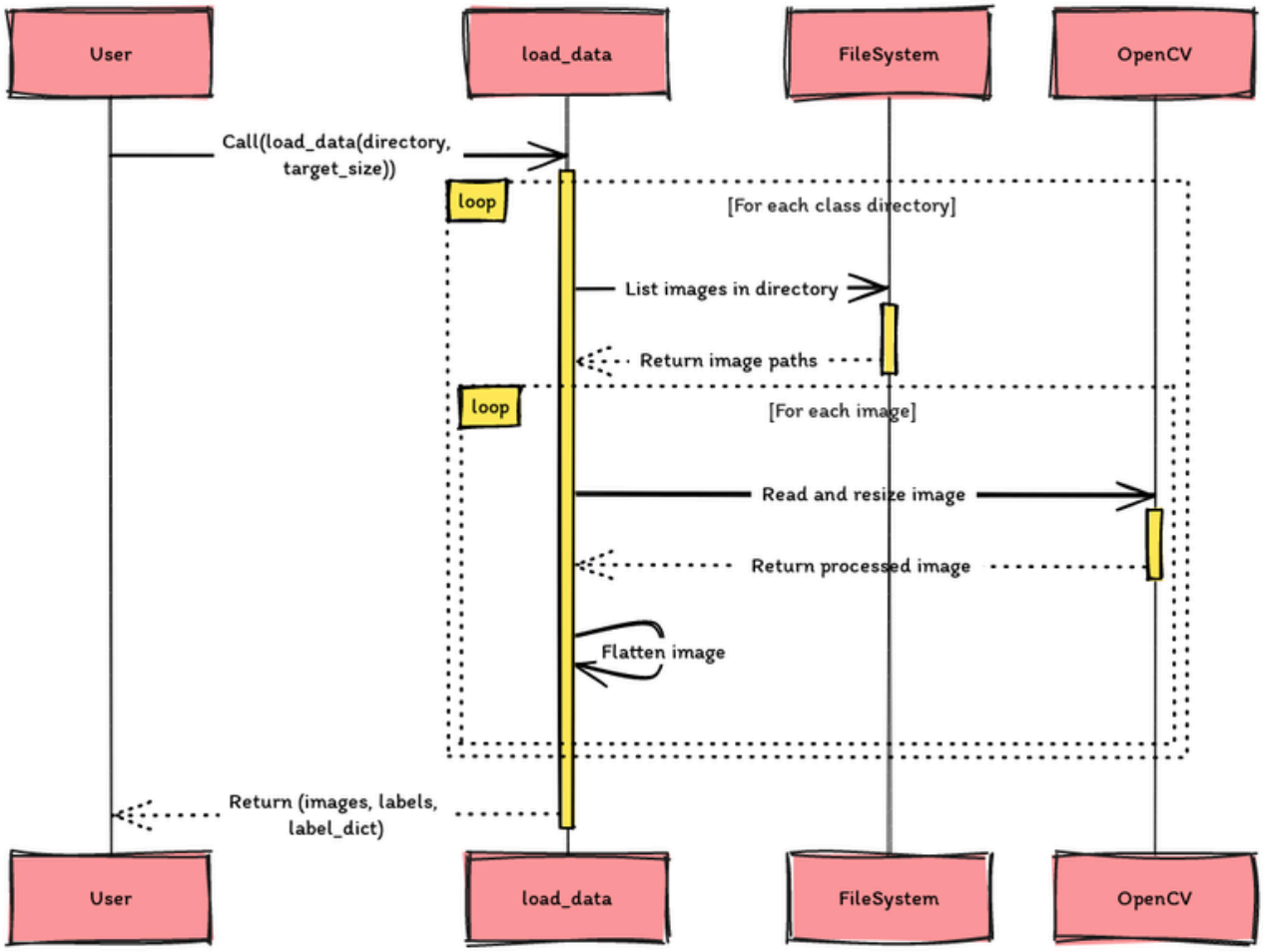


3.1 Data Pre-processing

In the data preprocessing stage of the American Sign Language (ASL) recognition project, each image endures a series of critical transformations to prepare it for effective feature extraction and subsequent classification. Firstly, images are converted from color (RGB) to grayscale. This reduction to a single channel diminishes the computational load and shifts the model's focus to the structural content of the images, rather than their color. Grayscale conversion is essential for highlighting the intensity of pixels which emphasizes the shapes and movements crucial for interpreting sign language gestures.

Following the color reduction, all images are resized to a uniform dimension of 64x64 pixels. This standardization simplifies the model's training and prediction processes by ensuring that each input into the model is consistent, both in shape and size. Such uniformity is vital because it allows the model to consistently learn from a standardized set of features across all images, which in turn facilitates more efficient learning and faster predictions.

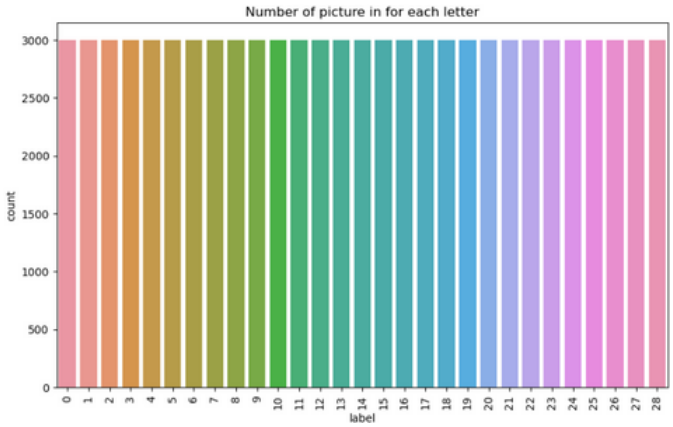
These preprocessing steps are integral to preparing the images for the next stage—feature extraction. Here, the images, now in grayscale and resized, are flattened from a 2D array into a 1D array consisting of 4096 features. This transformation is crucial as it converts the spatial information of the images into a numerical format suitable for processing by the decision tree classifier. By transforming the image data into a consistent, flattened format, the model can effectively analyze each pixel's intensity individually, enhancing its ability to recognize and classify intricate patterns specific to ASL. Through meticulous preprocessing, the data is optimally prepared, enhancing both the efficiency and accuracy of the model's performance in recognizing sign language.



3.2 EXPLORATORY DATA ANALYSIS (EDA)

Exploratory Data Analysis (EDA) plays a pivotal role in the preprocessing phase of the American Sign Language (ASL) recognition project. This analytical process involves a thorough inspection of the dataset to understand the distribution of the various classes and to visualize sample images. EDA is crucial for assessing the balance and integrity of the data before it enters the modeling stage.

During EDA, the distribution of classes is closely examined. This step is essential because it reveals how evenly the data is distributed among the different ASL signs represented in the dataset. If certain classes are overrepresented or underrepresented, it could lead to potential biases in the training of the model, where the model might perform well on more frequent classes but poorly on less frequent ones. Such imbalances can significantly skew the model's training and its subsequent predictive accuracy, as the model might not learn to generalize well across all classes.



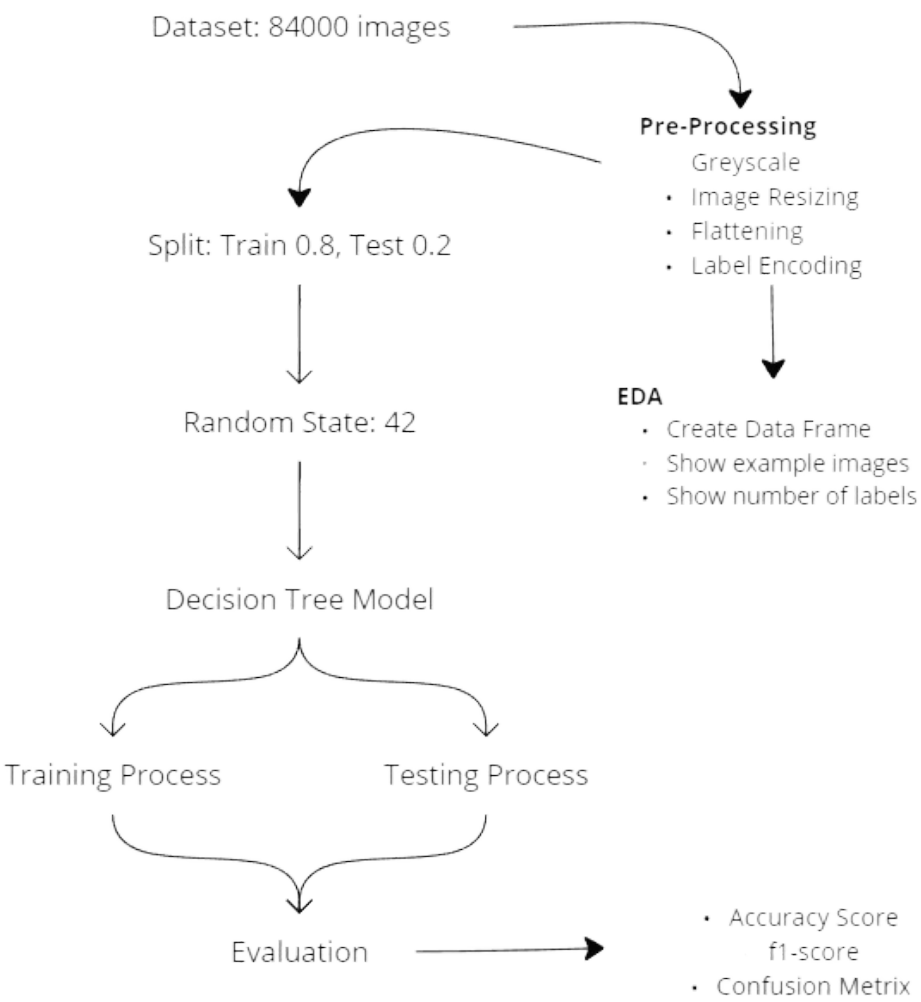
Visualization of sample images is another important aspect of EDA. By visually inspecting a subset of the dataset, researchers can confirm the quality and consistency of the images. This includes checking for any anomalies or outliers in the images, such as images that are too dark, too bright, or that do not correctly represent the intended ASL sign. Visualization also helps in understanding how well the preprocessing steps like resizing and grayscale conversion have been executed, ensuring that these transformations maintain the essential features necessary for accurate sign language recognition.

EDA provides a foundation for making informed decisions on further data preprocessing needs and model adjustments. By ensuring data balance and integrity through EDA, the project sets the stage for building a robust model capable of high performance and fairness in ASL sign recognition. This stage is integral to developing a machine learning solution that is reliable and effective across various real-world applications.

3.3 MODEL ARCHITECTURE

At the core of the model architecture is the Decision Tree Classifier, which operates by creating a tree-like structure of decisions. In this structure, nodes represent the points where data is split based on feature values, branches indicate the outcomes of these decisions, and leaf nodes denote the final classification results. The decision tree is trained on these extracted features, learning where best to split data at each node to optimize classification accuracy and reduce errors. To prevent the model from overfitting and to enhance its generalizability, the decision tree may undergo pruning, which involves removing less impactful sections of the tree.

The final stage involves evaluating the decision tree's performance through metrics such as accuracy, precision, and recall, which are crucial for assessing how well the model predicts new data. This model architecture is especially beneficial for the ASL recognition project due to its straightforward yet effective approach, which provides clear, interpretable results. Decision trees handle high-dimensional data efficiently, making them ideal for image classification where each pixel represents a feature. Moreover, their non-parametric nature ensures they do not assume any underlying data distribution, allowing them to adapt flexibly to various data types encountered in real-world applications.



3.3.1 DECISION TREE

A Decision Tree is a versatile supervised learning algorithm that is widely used for both classification and regression tasks. It constructs a model in the form of a tree structure, breaking down data sets into smaller subsets while simultaneously developing an associated decision tree. The tree consists of decision nodes, which represent the "questions" or tests on specific attributes, and leaf nodes, which are the outcomes or final decisions. This method is valued for its simplicity and interpretability, as it does not require statistical knowledge to understand and can be visualized easily, making it accessible for users from various fields.

The final stage involves evaluating the decision tree's performance through metrics such as accuracy, precision, and recall, which are crucial for assessing how well the model predicts new data. This model architecture is especially beneficial for the ASL recognition project due to its straightforward yet effective approach, which provides clear, interpretable results. Decision trees handle high-dimensional data efficiently, making them ideal for image classification where each pixel represents a feature. Moreover, their non-parametric nature ensures they do not assume any underlying data distribution, allowing them to adapt flexibly to various data types encountered in real-world applications.

Decision trees require minimal data preparation compared to many other data techniques, not necessitating normalization or scaling of data. They can handle both numerical and categorical data and are particularly effective for large datasets. However, they are prone to overfitting and can be sensitive to small changes in the data, which may lead to a different set of splits. Decision trees can also exhibit bias toward classes that are more frequent, hence why balancing the dataset before training is generally recommended.

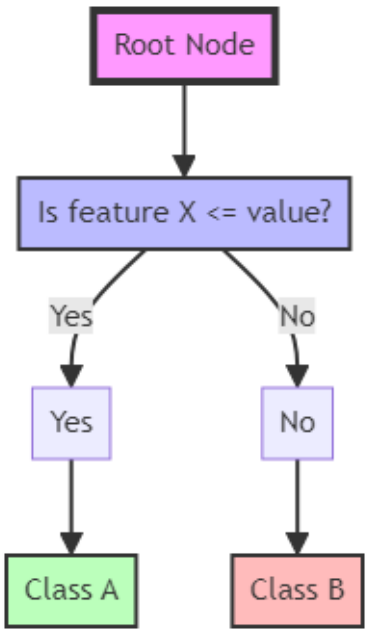
Despite these works, decision trees are fundamental components of more complex algorithms like Random Forests and Gradient Boosting Machines, underscoring their utility and scalability. They remain a popular choice due to their straightforwardness and the clear, logical model structure they provide, which is crucial in scenarios requiring transparency and easy interpretation of the model's decision-making process.

Measure	**Value**
max_depth	None
max_features	None
min_smples_leaf	5
Splitter	best
min_weight_fraction_leaf	0

3.3.2 TRAINING A DECISION TREE FOR HAND IMAGES

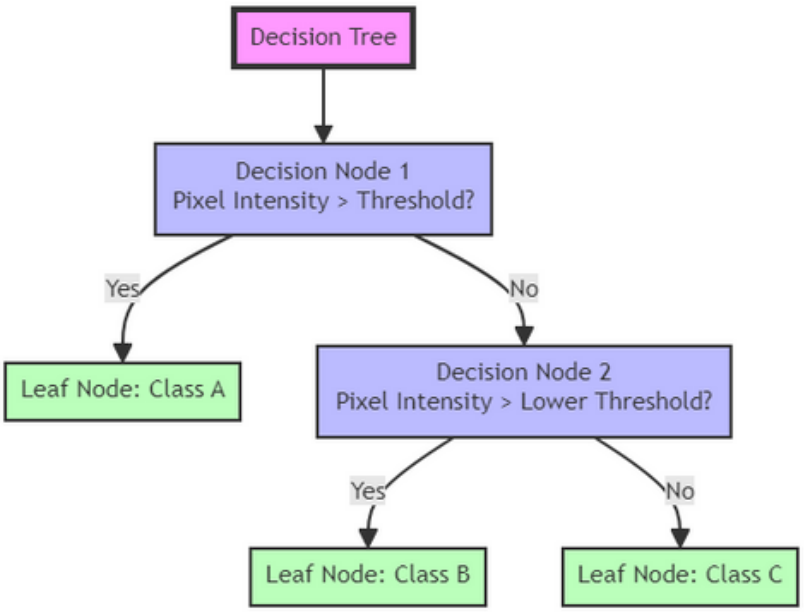
The first step is to gather a comprehensive collection of images depicting ASL hand signs. These images need to capture a wide range of variations in hand position, orientation, and shape. It's important to include pictures taken under different lighting conditions and against various backgrounds to mimic the diversity of real-world environments. This diversity helps train a model that is not just accurate under ideal conditions but is robust enough to handle less-than-perfect real-life scenarios.

Once the dataset is prepared, the next step involves extracting meaningful features from each image that the decision tree will use for making classifications. Key features typically include pixel intensity and edge detection.



In the training process, the decision tree algorithm will look for ways to best split the data based on the features. For example, it might find that certain pixel intensities or specific edge patterns are good at differentiating between the sign for 'A' and 'B'. The tree essentially learns a series of "if-then" rules that help it decide what sign an image is showing based on the features of that image.

In a decision tree, Decision Nodes are crucial points where the tree makes splits in the data. Each node represents a question about a specific feature, such as whether the intensity of a particular pixel is above a certain threshold. Based on the answers, the dataset is divided accordingly, guiding the path down the tree. Leaf Nodes, on the other hand, are the endpoints of these paths. No further splits occur at these nodes, and each one assigns a class label to the input data it receives. In the context of recognizing hand signs, each leaf node would assign a label that corresponds to a specific hand sign, effectively determining what sign is represented by the image data it analyzed. This structure allows the decision tree to categorize images of hand signs accurately by following a series of simple decision-making steps.



3.3.3 MODEL EVALUATION

Our model achieves an overall accuracy of 91.63%, indicating a high rate of correct predictions across all sign categories. This metric confirms that the decision tree has effectively captured the essential patterns and features necessary for ASL sign recognition.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FN+FP}$$

Further insights are gained through the classification report, which breaks down the precision, recall, and F1-score for each sign:

- Precision measures the accuracy of positive predictions. It reflects the percentage of results which are relevant to the sign being tested.
- Recall(Sensitivity) assesses the model's ability to correctly identify all relevant instances of a sign. It is particularly important in applications like ASL recognition, where missing a sign can lead to misunderstandings.

$$Sensitivity = \frac{TP}{TP + FN}$$

- F1-score is the harmonic mean of precision and recall and provides a single metric that balances both the precision and the recall.

$$\text{F score} = 2 * \frac{\text{Precision} * \text{sensitivity}}{\text{Precision} + \text{sensitivity}},$$

Class Label	Precision	Recall	F1-Score	Support
0	0.85	0.88	0.86	584
1	0.83	0.84	0.84	574
2	0.91	0.88	0.89	617
3	0.89	0.88	0.89	589
4	0.95	0.95	0.95	602
...	...	...	...	...
28	0.92	0.93	0.93	579
Overall	0.92	0.92	0.92	17400

In evaluating the model, we consider three crucial metrics: precision, recall, and F1-score, particularly looking at their performance for class 4. Precision is a measure of the accuracy of positive predictions made by the model; for class 4, it stands at 0.95, meaning that 95% of the predictions the model made for class 4 are correct. Similarly, recall or sensitivity for class 4 is also 0.95, which shows that the model successfully identifies 95% of all actual class 4 instances in the dataset. The F1-score, which is a weighted average of precision and recall, also reaches 0.95 for class 4.

Classes such as 4, 8, and 15 are examples of high-performing classes, where the model achieves very high precision, recall, and F1-scores. This indicates that the model is particularly effective at predicting these signs accurately, consistently distinguishing them from others with minimal error.

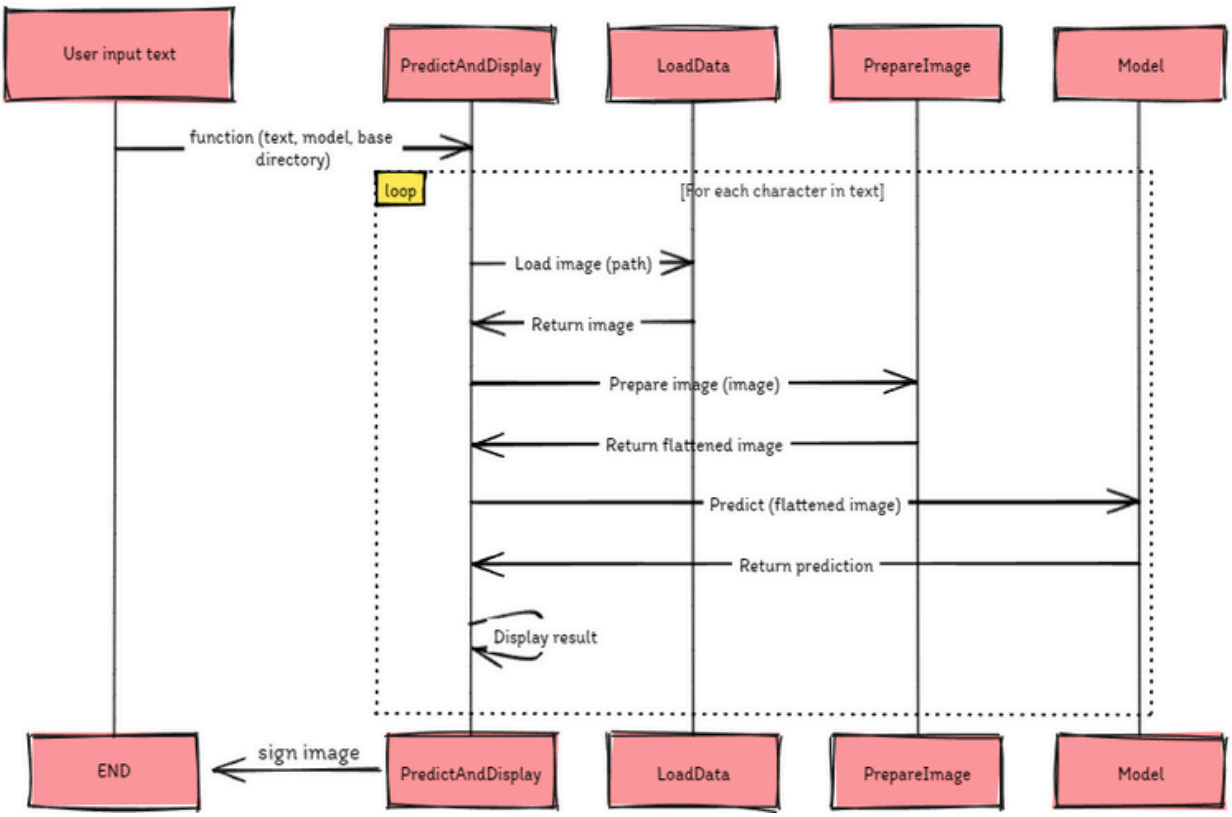
On the other hand, there are classes that pose challenges for the model, as evidenced by their lower scores in one or more evaluation metrics. For example, class 24 demonstrates lower precision and F1-score compared to other classes. This suggests a relatively higher rate of false positives for class 24, indicating that while the model often predicts this class, it is not always correct in its predictions.

3.3.4 CREATE FUNCTION TO CONVERT TEXT TO SIGN LANGUAGE

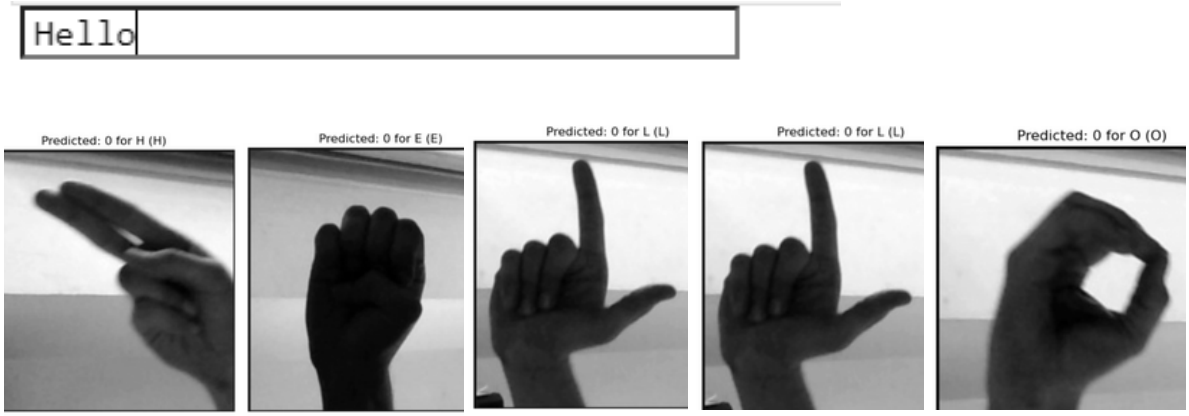
In the programming section dedicated to translating text to sign language, the project utilizes a series of functions designed to interface with a pre-trained machine learning model. I create the function for responsible for reading images from a specified path and converting them to grayscale, which simplifies the model's task by focusing on the shape and texture rather than color information.

I create function takes these grayscale images and resizes them to a uniform size, then flattens the 2D image arrays into 1D feature vectors. This standardization is crucial for the model to process the data consistently.

The core function will iterates over each character in the input text, finding corresponding images in a predefined directory structure. It gracefully handles errors like missing directories or images, ensuring robustness. Upon successful image retrieval, the function processes the image through the aforementioned steps and feeds the prepared data into the model for prediction. It then displays the image with a title indicating the predicted sign language gesture corresponding to the text character.



In the practical application of the text-to-sign language translation system, users begin by entering text they wish to translate into sign language. This text is processed through a set of functions designed to convert each character into its corresponding sign language gesture using a decision tree classifier



4.1 CONCLUSION

Our project has successfully developed a sign language recognition app, which is a significant advancement for the deaf and hard-of-hearing community. Utilizing Decision Trees, this application interprets sign language, enhancing communication, education, and social interaction for its users. Moving forward, we aim to enrich the app's capabilities by enabling it to comprehend full phrases and sentences. Additionally, we plan to integrate more advanced machine learning technologies to improve its efficiency and user-friendliness. This endeavor illustrates the transformative potential of machine learning in enhancing communication accessibility.

4.2 RECOMMENDATIONS AND FUTURE RESEARCH

While the Decision Tree model has achieved an accuracy of 91.63%, exploring other models such as Neural networks, particularly Convolutional Neural Networks (CNNs), which are renowned for their effectiveness in image recognition tasks, could enhance our app's performance and potentially offer higher accuracy.

Expanding our app to include various sign languages such as Thai and Chinese, as well as signs that represent whole words or phrases, could make it more versatile and useful for a global audience. Enhancing the app's ability to understand the nuances of conversation could improve its usefulness in practical communication scenarios.

Developing the app to support two-way communication would make it a more powerful tool, enabling not just interpretation but also conversation between users. These enhancements aim to leverage cutting-edge technology to break down communication barriers and create a more inclusive environment for individuals who use sign language.

REFERENCES

Chang, (2013). A Decision Tree based Real-time Hand Gesture Recognition Method using Kinect. Journal of Korea Multimedia Society. Retrieved from
http://koreascience.or.kr/article/JAKO201303537263548.page

Patidar, (2011). Sign Gesture Recognition using Combined Features of Sum graph and HMM with Decision tree classifier. Retrieved from
https://ijcsit.com/docs/Volume%202/vol2issue3/ijcsit2011020368.pdf

Shibata, (2016). Basic investigation for improvement of sign language recognition using classification scheme. Retrieved from
https://link.springer.com/chapter/10.1007/978-3-319-40349-6_55

Chavan, A., Deshmukh, S., & Fernandes, F. (2022). Sign Language Detection. Retrieved from
https://arxiv.org/pdf/2209.03578

Vijitkunsawat, W., Racharak, T., Nguyen, C., & Le Minh, N. (2023). Video-Based Sign Language Digit Recognition for the Thai Language: A New Dataset and Method Comparisons. Retrieved from
https://www.scitepress.org/Papers/2023/116437/116437.pdf

Ahli, M. E. M. (2023). Towards a Reliable Machine Learning-based Model Designed for Translating Sign Language Videos to Text. Retrieved from
https://repository.rit.edu/cgi/viewcontent.cgi?article=12625&context=theses

You, H., Kang, H., & Lee, S. (2023). Efficient skin segmentation model for sign language recognition. Mathematics, MDPI. Retrieved from
https://www.mdpi.com/2227-7390/11/9/2057

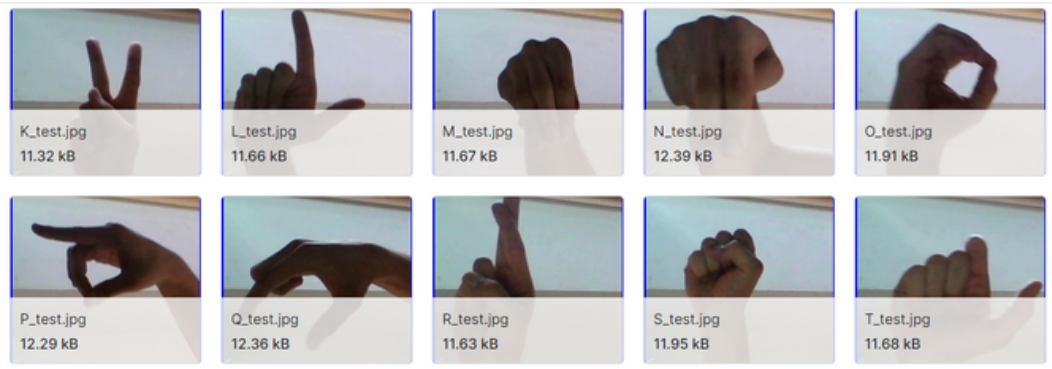
Kolkur, S., Kalbande, D., Shimpi, P., Bapat, C., & Jatakia, J. (2017). Human Skin Detection Using RGB, HSV and YCbCr Color Models. Retrieved from
https://arxiv.org/pdf/1708.02694

Appendix

Go to see my full notebook: https://github.com/rachata072/Development-of-a-Sign-Language-Dictionary-App-Using-Supervised-Learning/blob/main/SignLang.ipynb

Example of Dataset

Source: https://www.kaggle.com/datasets/grassknoted/asl-alphabet



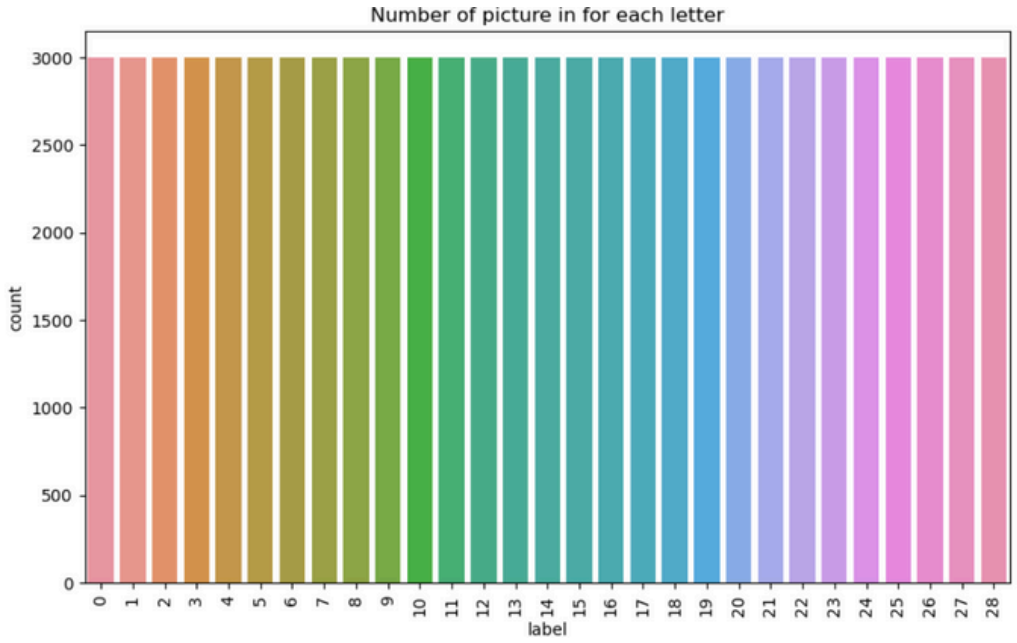
Example of Code

Load and pre processing data:
1. Grayscale Conversion
3. Flattening
4. Label Encoding

```
#create function to load data set and generate size of images
def load_data(directory, target_size=(64, 64)):
    labels = []
    images = []
    label_dict = {name: num for num, name in enumerate(os.listdir(directory))}
    for label in tqdm(os.listdir(directory)):
        class_path = os.path.join(directory, label)
        for img_path in os.listdir(class_path):
            img = cv2.imread(os.path.join(class_path, img_path), cv2.IMREAD_GRAYSCALE)
            img = cv2.resize(img, target_size)  # resize image
            img = img.flatten()  # flatten the image from 2D to 1D
            images.append(img)
            labels.append(label_dict[label])
    images = np.array(images)
    labels = np.array(labels)
    return images, labels, label_dict
```

EDA to plot for check number of images in each labels

```
plt.figure(figsize=(10, 6))
sns.countplot(x='label', data=df)

plt.title("Number of picture in for each letter")
plt.xticks(rotation=90)
plt.show()
```

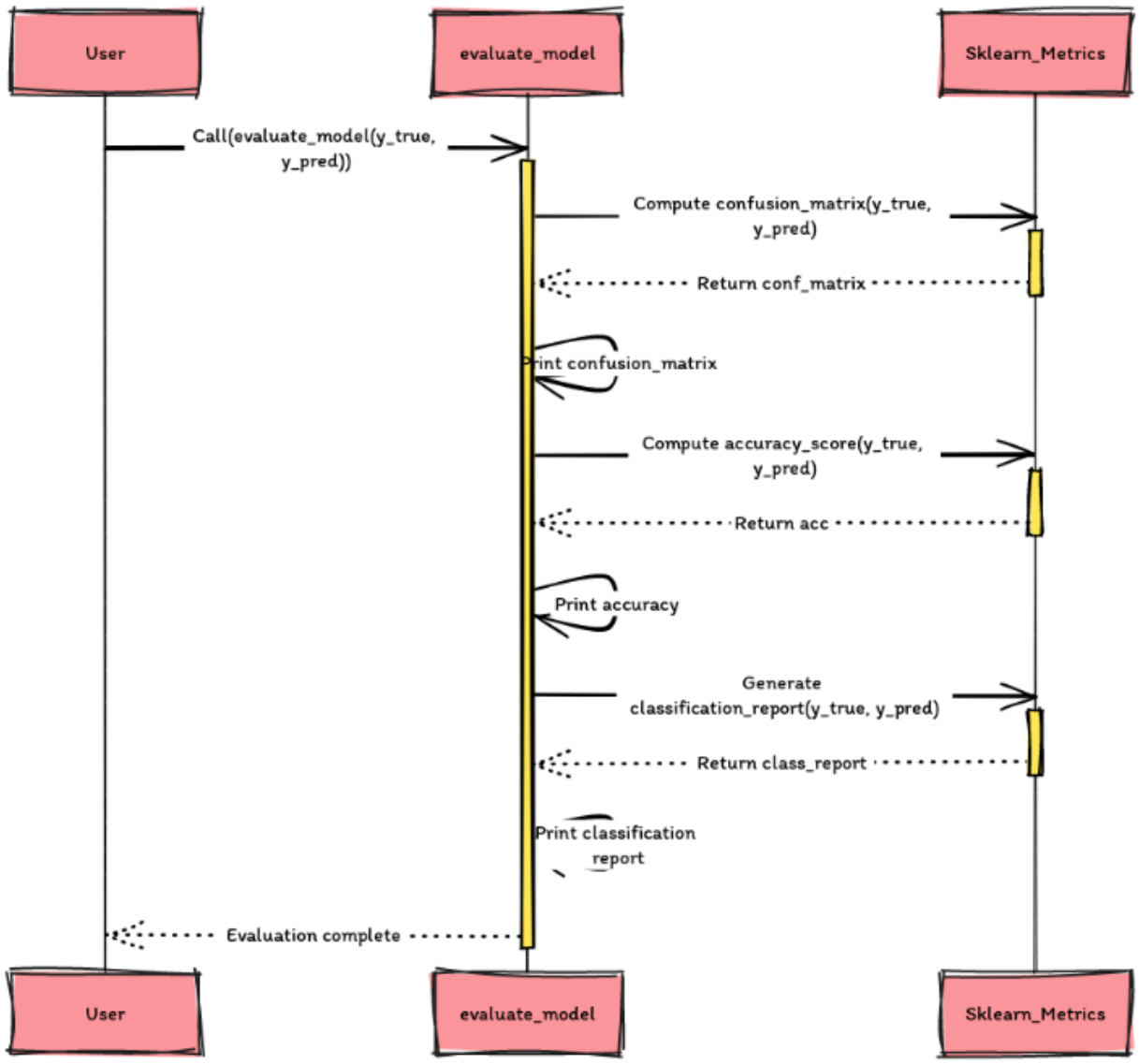


Build function to show example of data after transform randomly

```
def display_sample_images(df, num_images=5):
    sample_images = df.sample(n=num_images, random_state=42)
    fig, axes = plt.subplots(1, num_images, figsize=(10, 2))

    for i, ax in enumerate(axes.flat):
        # Reshape the flattened pixels back to 2D array
        img = sample_images.iloc[i][:-1].values.reshape(64, 64)
        ax.imshow(img, cmap='gray')
        ax.set_title(f"Label: {sample_images.iloc[i]['label']}")
        ax.axis('off')

    plt.tight_layout()
    plt.show()

# Display sample images from the DataFrame
display_sample_images(df)
```

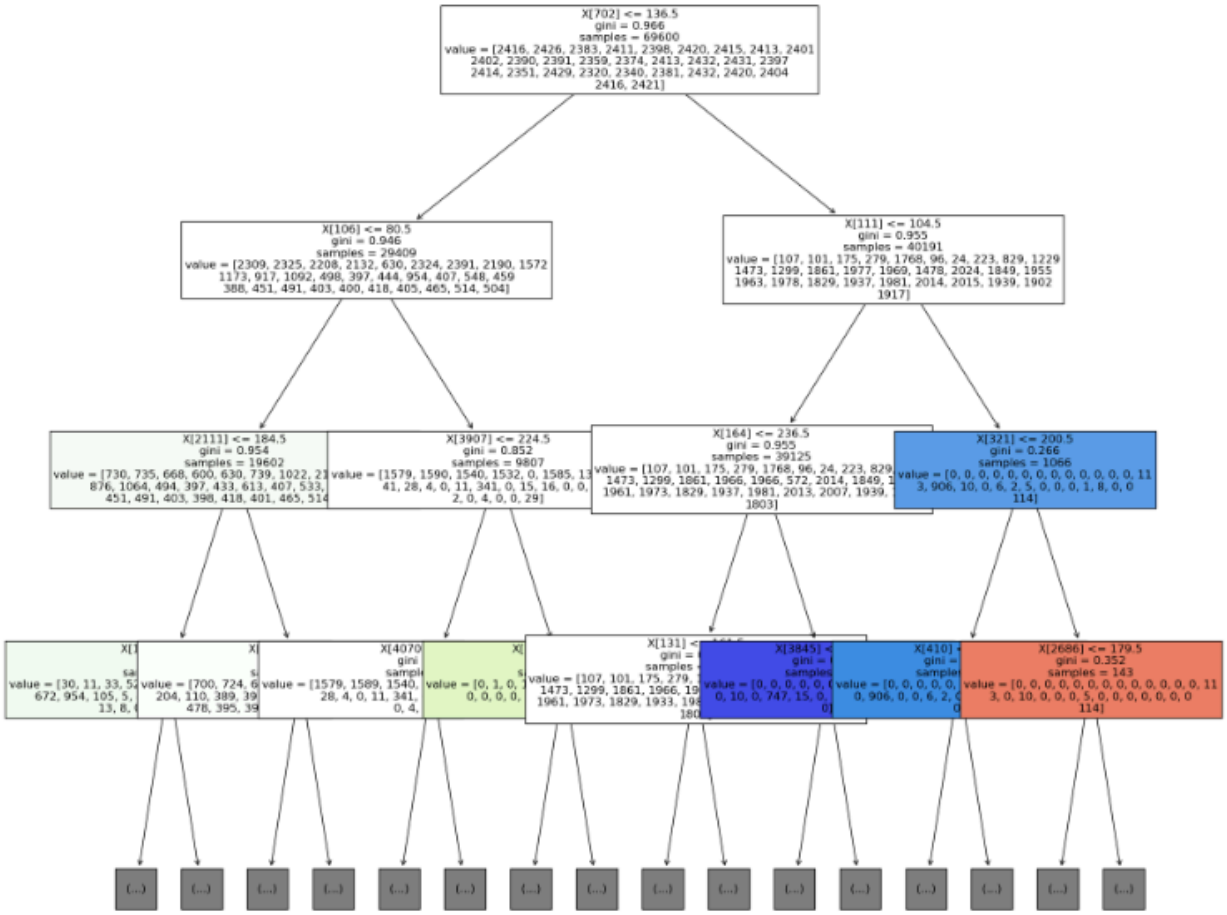
Build function to evaluate model by confusion metrix, accuracy_score, and report

```python
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

def evaluate_model(y_true, y_pred):
    # Compute the confusion matrix
    conf_matrix = confusion_matrix(y_true, y_pred)
    print("Confusion Matrix:\n", conf_matrix)

    # Calculate the overall accuracy
    acc = accuracy_score(y_true, y_pred)
    print("Overall accuracy:", acc)

    # Generate a classification report
    class_report = classification_report(y_true, y_pred)
    print("Classification Report:\n", class_report)
```



To plot decision tree graph

```
plt.figure(figsize=(20,20))
plot_tree(dt_classifier, filled=True, max_depth=3, fontsize=12)
plt.show()
```



Build function to convert text inputed to sign language image

```
def prepare_image(image, target_size=(64, 64)):
    if image is None:
        raise ValueError("Invalid image provided")
    image = cv2.resize(image, target_size)
    return image.flatten()

def predict_and_display(text, model, base_directory):
    for char in text.upper():
        directory_name = "space" if char == " " else char
        img_directory = os.path.join(base_directory, directory_name)

        if not os.path.exists(img_directory):
            print(f"Directory for '{char}' does not exist. Expected at: {img_directory}")
            continue

        image_files = os.listdir(img_directory)
        if not image_files:
            print(f"No images available for '{char}' in {img_directory}")
            continue

        image_file = image_files[0]
        full_path = os.path.join(img_directory, image_file)

        try:
            image = load_data(full_path)
            prepared_image = prepare_image(image)
            prediction = model.predict([prepared_image])
            predicted_label = np.argmax(prediction)

            plt.figure()
            plt.imshow(image, cmap='gray')
            plt.title(f"Predicted: {predicted_label} for {char} ({directory_name})")
            plt.axis('off')
            plt.show()
        except Exception as e:
            print(f"An error occurred while processing {full_path}: {e}")
```

```
base_directory = "my-path"

model = dt_classifier
text_input = input()
predict_and_display(text_input, model, base_directory)
```