**CSCI B500 ASL Sign Language Detector**

**Shourish Kothawale**

**Abstract:**

Sign language is a crucial form of communication for individuals who are deaf or hard of hearing, enabling the expression of thoughts and emotions. Automating the classification of American Sign Language (ASL) letters using techniques like Convolutional Neural Networks (CNNs) can enhance accessibility tools. This project aims to classify ASL letters using deep learning models, utilizing the Sign Language MNIST dataset from Kaggle, which contains 200x200 pixel images of hand gestures representing ASL letters. Each image is labeled with its corresponding letter, making it ideal for training classification models.

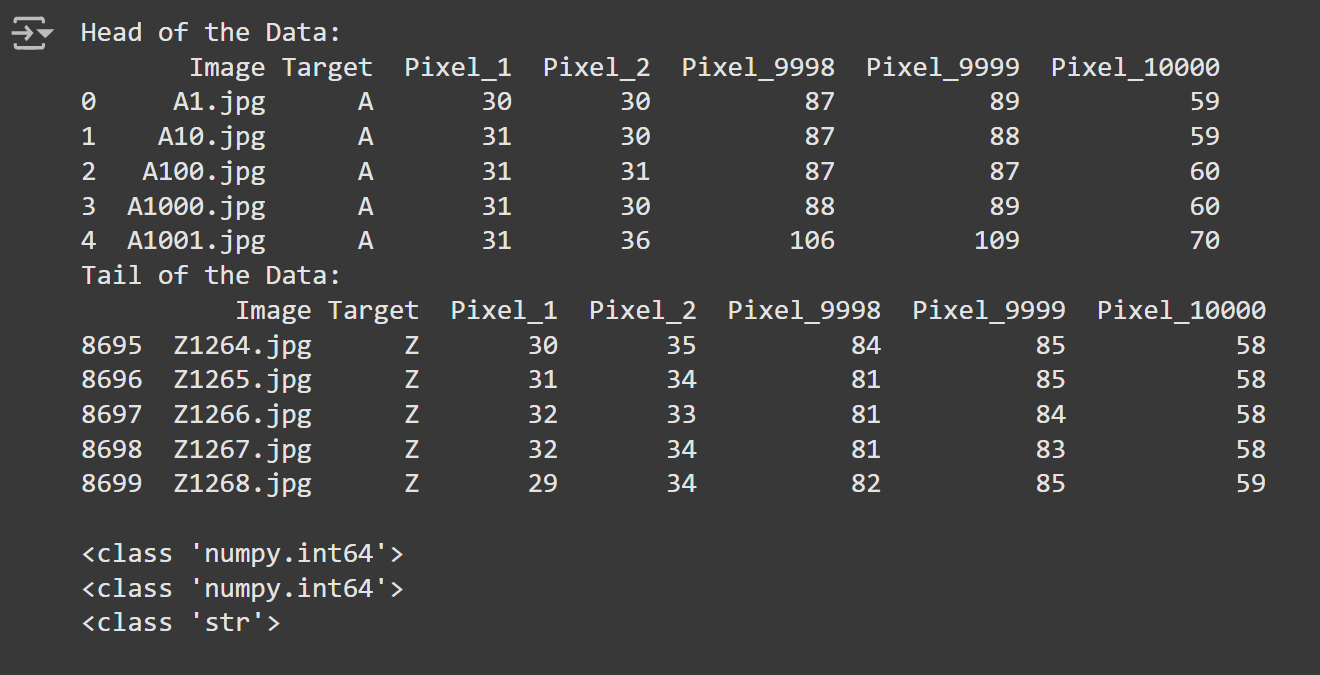
The project begins with Exploratory Data Analysis (EDA) to understand the dataset, followed by data munging for preprocessing and transformation. We analyze the dataset's statistical properties to identify patterns, then implement and evaluate CNNs and other techniques for ASL classification, comparing their performance. This approach ensures a comprehensive understanding of the dataset while optimizing algorithms for accurate classification of ASL letters.

**Exploratory Data Analysis (EDA):**

In this section, we will focus on Exploratory Data Analysis (EDA) and data transformation to prepare our dataset for effective model building. EDA serves as a critical step in understanding the dataset, helping us uncover patterns, relationships, and potential anomalies in the data. By thoroughly exploring the data, we can gain valuable insights into its structure and distribution, which will guide subsequent steps in our workflow.

We will perform various exploratory tasks, such as visualizing the dataset through plots and charts to better understand the frequency and representation of different classes. These visualizations will help us identify any imbalances or peculiarities in the data. Additionally, we will extract key statistics to summarize the dataset, offering a clearer view of its properties.

The data transformation process will focus on preparing the dataset for analysis and modeling. This includes cleaning, normalizing, and reshaping the data to ensure compatibility with our machine learning pipelines. These transformations not only make the data easier to work with but also optimize it for performance during the training and evaluation phases. Through this dual approach of exploration and transformation, we aim to build a solid foundation for achieving accurate and reliable classification results.

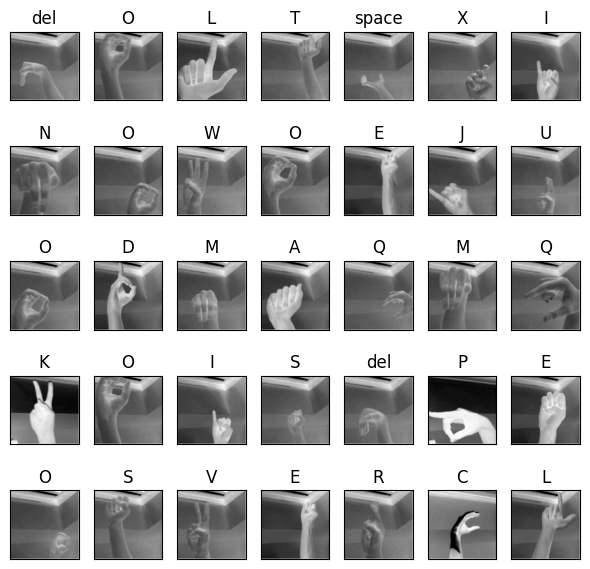


We begin by examining the dataset's structure, which consists of three main components: an image column (containing file names), a target column (representing the ASL letter), and pixel features (10,000 individual attributes representing pixel intensities in grayscale from 0 to 255). After cleaning, we omit the image and index columns as they do not contribute to the analysis. The remaining columns are:

* **Target (str)**: Represents the ASL letter.
* **Pixel\_n (int)**: Pixel values in the range of 0 to 255, corresponding to the image's grayscale intensity.

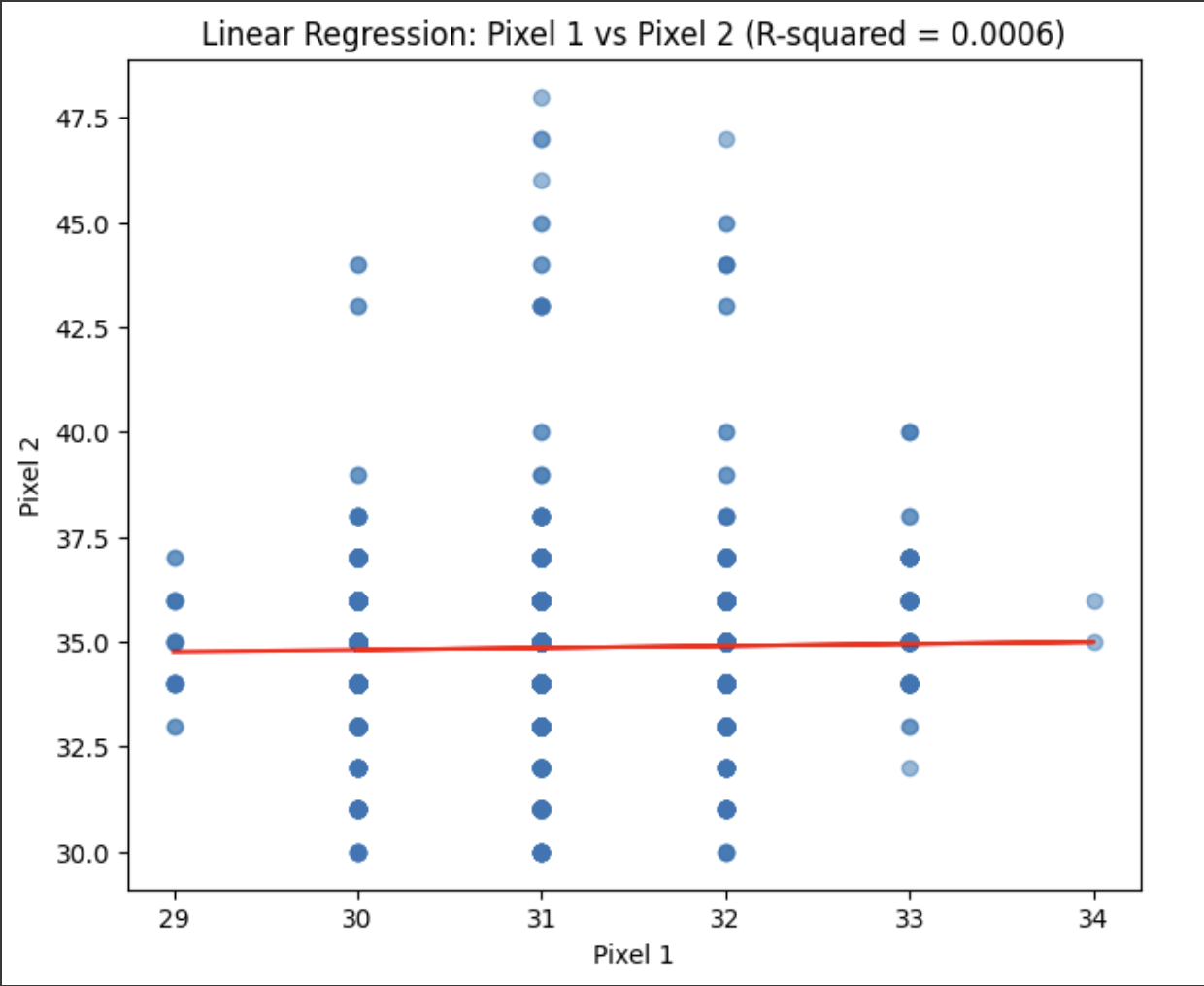
We proceed with Exploratory Data Analysis (EDA), analyzing metrics like mean, median, mode, max, and min to understand the data's distribution and edge cases. To streamline our workflow, we convert the data into a NumPy ndarray, which facilitates data manipulation and ensures compatibility with machine learning libraries. It's also essential to separate the target column from the pixel features to guide the model's training process.

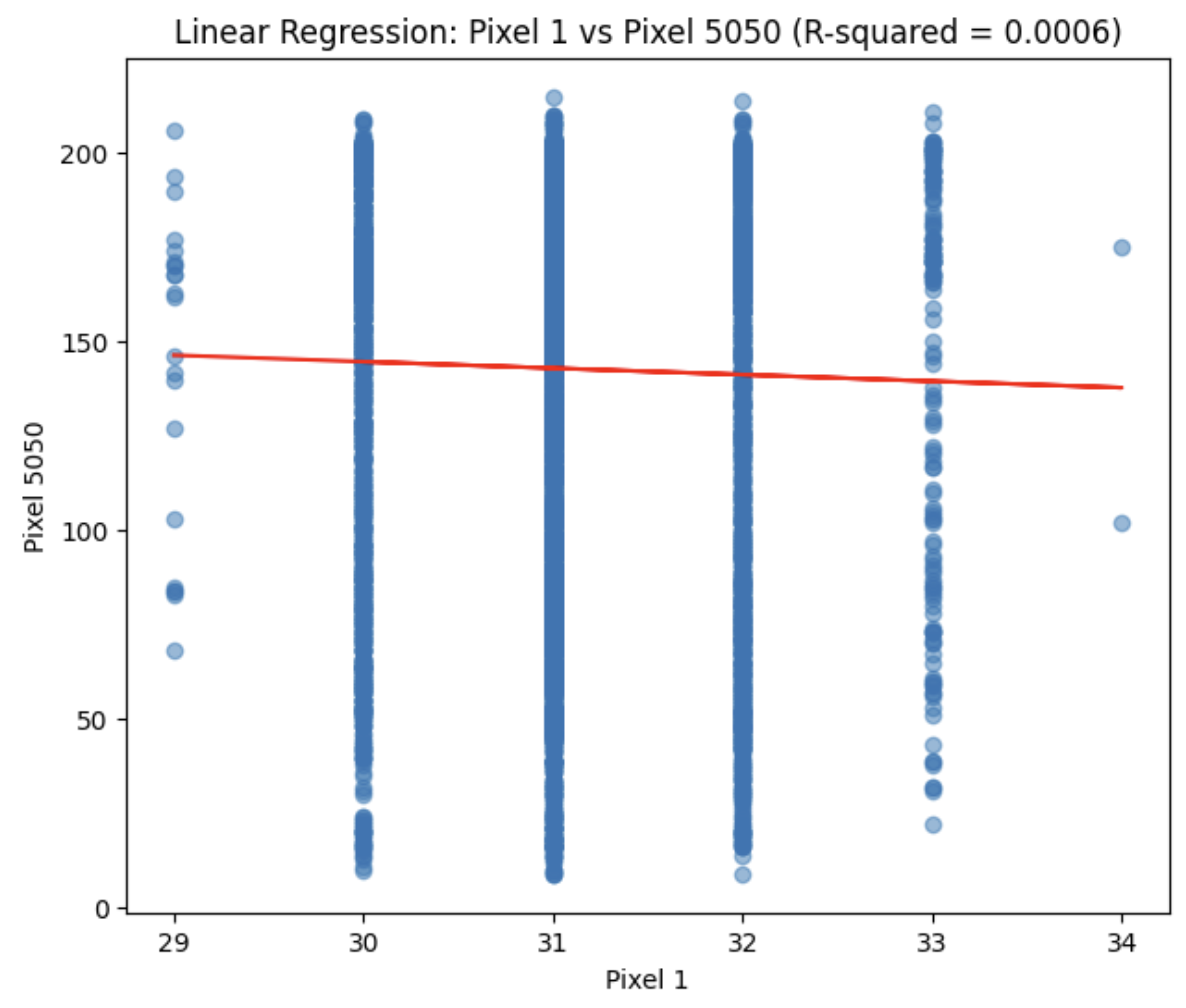
Next, we visualize the data by creating an image plot that randomly selects and displays images from the dataset. This plot helps verify the preprocessing steps, showing that the data has been converted to grayscale and reduced in fidelity. These transformations simplify the data while preserving key features, ensuring it is ready for machine learning analysis.



The **image plot** displays 35 random images from the dataset, offering a quick overview of the data's visual characteristics. Despite the reduced fidelity from preprocessing, the images remain interpretable, making it easy to identify the ASL letters. However, human readability does not ensure high accuracy for machine learning models. The reduced fidelity and grayscale conversion may impact the model's performance, depending on how well the relevant features for classification are preserved. This visualization confirms the data's usability, though further steps like feature engineering and model tuning are needed for accurate evaluation.

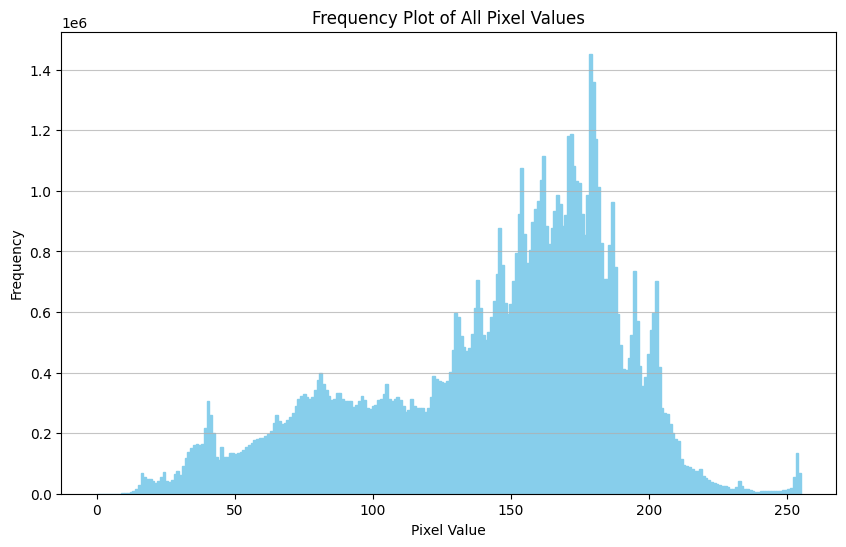
Next, we apply **linear regression** to examine the relationship between neighboring and distant pixels. While not highly informative for classification, it helps us understand potential correlations between pixel intensities. By testing the relationships between adjacent and far-apart pixels, we gain insights into spatial dependencies within the image data. Although this may not directly improve accuracy, it offers a perspective on the structure of the data.





The **image plots** provide a visual overview of the data, though they do not offer significant insights into pixel relationships. The correlation coefficients between pixel values are close to zero, suggesting minimal correlation. However, we observe vertical alignments in the plots, likely due to the discrete nature of pixel intensities, which are quantized into limited values.

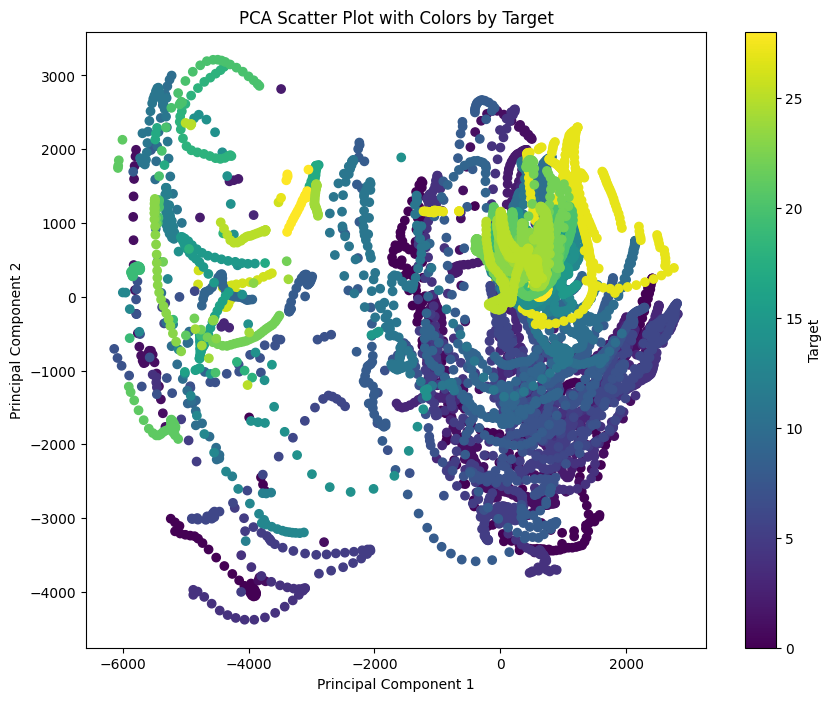
To better understand pixel distribution, we can create a **frequency plot** to show how often each pixel intensity appears across the images. This helps identify common pixel values and their locations. Additionally, we can use a **histogram** to visualize the distribution of pixel intensities, revealing whether the images are concentrated in specific intensity ranges or if they cover the full grayscale spectrum. This analysis can inform preprocessing and model development, helping improve model performance.



From the histogram, we can clearly observe the composition of pixel values across the dataset. We notice that pixel values ranging from 125 to 200 appear more frequently in the images. This is likely due to the hand itself, which tends to be brighter than the background, as we saw in the earlier image plots. These pixel intensities represent the lighter areas of the image, corresponding to the hand gestures in ASL.

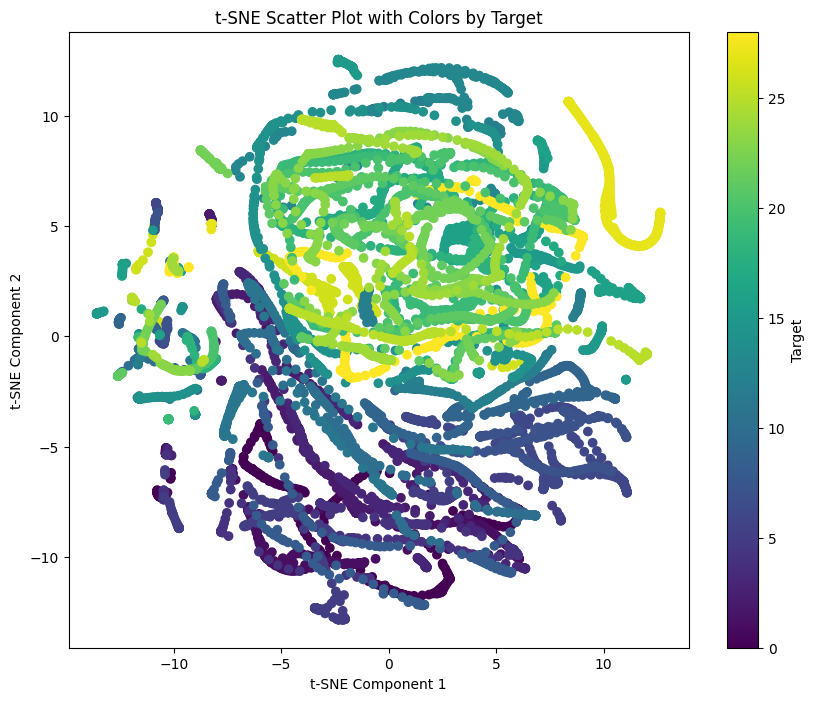
To gain further insights into the variability and structure of the data, we can apply **dimensionality reduction** techniques. These methods allow us to reduce the high-dimensional pixel data to a lower-dimensional space, making it easier to visualize and interpret. By performing dimensionality reduction, we can better understand the variation in the data and potentially identify clusters or groups within the dataset.

Principal Component Analysis (**PCA**) and **t-SNE** (t-Distributed Stochastic Neighbor Embedding) are popular dimensionality reduction techniques that are well-suited for this task. PCA helps us identify the most important features (principal components) that explain the variance in the data, while t-SNE is particularly useful for visualizing high-dimensional data in two or three dimensions, often revealing underlying patterns or groupings.



The **PCA plot** shows that the data appears scattered, with no discernible patterns or groupings. This suggests that the first two principal components (PC1 and PC2) fail to capture the underlying structure of the data, explaining only 38.87% of the total variability (24.92% from PC1 and 13.95% from PC2). This limited explanation of the variance indicates that PCA, which focuses on capturing global variance, is not effective in revealing the true complexity of the dataset.

To address this, we turn to **t-SNE** (t-Distributed Stochastic Neighbor Embedding), a more powerful technique for visualizing high-dimensional data. t-SNE is better at preserving local structure and clustering similar data points, which can reveal hidden patterns and groupings that PCA might overlook. By applying t-SNE, we aim to gain a clearer understanding of how the different ASL letters are related, potentially revealing more distinct clusters or separations between the classes. This will help us uncover insights into the structure of the data that may improve model performance and inform further analysis.



TSNE just like its counterpart has similar problems. This is evident since there are **10,000** dimensions, and it would be a very hard to capture them in 2 PCs

This concludes the EDA for this data, we can now move into our Classification tasks with a Convolutionary Neural Network (CNN) and K-Nearest Neighbors Classifier

Despite using **t-SNE**, we face similar challenges as with PCA. With 10,000 dimensions in the dataset, it's difficult to capture meaningful patterns in just two dimensions. Both techniques struggle to represent the complexity of the high-dimensional pixel data, resulting in scattered plots.

Having completed **Exploratory Data Analysis (EDA)**, we now move on to **Data Munging**. In this phase, we'll clean and transform the data, handle missing values, scale or normalize features, and prepare the dataset for modeling. This step is crucial for ensuring the data is in the right format for machine learning algorithms.

**Data Munging/Wrangling:**

After completing the Exploratory Data Analysis (EDA), we transition to the crucial phase of **Data Munging**, which involves cleaning and transforming the raw data into a more usable format for analysis and machine learning. This step ensures that the data is properly preprocessed, making it suitable for training accurate models.

The data munging process in this project was carried out through Python scripts, addressing several key aspects. Initially, we reduced the size of the dataset. The original dataset contained 84,000 images, but for efficiency and to make the analysis more manageable, we reduced the dataset to 8,400 images. This was done by selecting a representative sample of the data, ensuring that the model still had a sufficient amount of information for training, while avoiding the computational burden of working with the entire dataset.

Next, we converted all the images to grayscale. The original dataset consisted of color images, but to simplify the data and focus on the core features, we transformed them into grayscale. This change significantly reduces the complexity of the data by eliminating color channels, but it still preserves the critical visual information, such as the intensity variations in the hand gestures, which are essential for ASL classification.

Lastly, we resized the images from their original 200x200 pixel resolution to a smaller, more manageable size of 100x100 pixels. While the original resolution provided a lot of detail, working with 200x200 images was computationally expensive. By resizing the images, we reduced the data size and improved processing efficiency, while still retaining enough detail for the machine learning models to detect and classify the hand gestures effectively.

These steps were crucial in preparing the dataset for machine learning. Through data munging, we ensured that the data was cleaned, transformed, and optimized, making it suitable for model training and analysis.

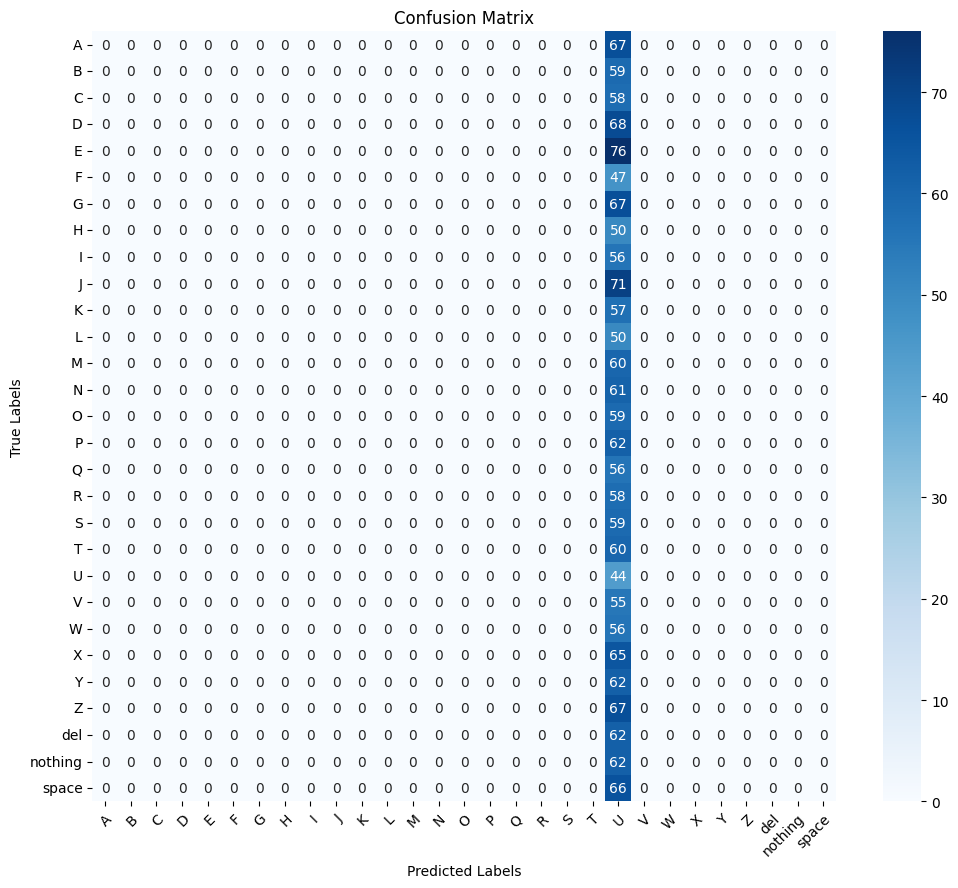
**Advanced Data Analysis and Results:**

A **Convolutional Neural Network (CNN)** is a deep learning model designed for processing grid-like data such as images. It uses convolutional filters to automatically extract features, like edges and textures, and processes them hierarchically for tasks like image classification.

We initially used a CNN to classify American Sign Language (ASL) gestures, expecting a straightforward task, but it turned out to be more complex. While understanding each layer's role was clear, hyperparameter tuning proved challenging due to the large number of parameters, requiring significant trial and error.

To improve efficiency, we considered using **transfer learning**, which allows leveraging pre-trained models like **MobileNetV2** or **ResNet**. These models, trained on large datasets, can be fine-tuned for our specific task, offering a quicker and potentially more accurate solution. However, transfer learning was not implemented in this project but remains a promising option for future improvements.

The confusion Matrix and the accuracy of the CNN trained was critically low as the layers need more hyperparameter tuning but due to time constraints the model was left as it is, with a **3%** accuracy. The following confusion Matrix shows how inaccurately the model only predicted one letter correctly which was **U**. One of the main reasons of this could be that the layers of the CNN aren’t well optimized, or there could be an error that is setting the model off.

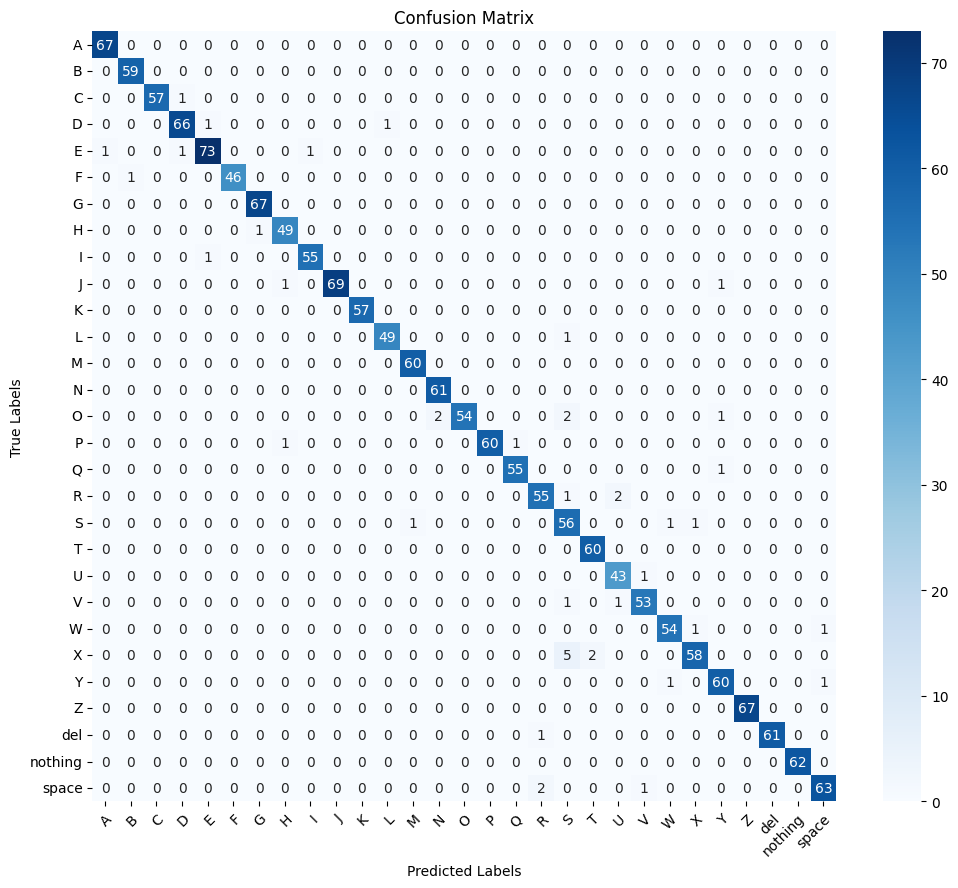


We implemented **K-Means clustering** to gain insights into the data and group similar features before applying classification. K-Means is an unsupervised learning algorithm that partitions the dataset into **K** distinct clusters based on feature similarity. This process helps identify patterns in the data, which can guide feature selection and improve model optimization.

To further enhance the model’s performance, we incorporated **K-Folds cross-validation** with K-Means. K-Folds splits the dataset into K subsets, or "folds." The model is trained on K-1 folds and validated on the remaining fold, repeating the process K times. This ensures that each data point is used for both training and validation, providing a more reliable performance estimate and reducing the risk of overfitting.

By using K-Folds cross-validation, we were able to efficiently perform **hyperparameter tuning**. The random sampling of the data through K-Folds helps identify the best model configuration. This approach resulted in achieving nearly **95% accuracy** with the model.

The effectiveness of this approach was further confirmed by the **confusion matrix**, which showed high precision and recall with minimal misclassifications, indicating that the model performed well in predicting the correct ASL letters.



The confusion matrix is nearly perfect as most of the values of the heatmap lie on the diagonal. But we must make sure that our model doesn’t overfit or else there are chances for the model to fail when it encounters new data.

**Future Works:**

Future work will focus on enhancing the ASL gesture classification model. One improvement is incorporating transfer learning by using pre-trained CNN models like MobileNetV2 or ResNet, which can boost accuracy and speed up training, especially with limited data. Another possibility is integrating the model with a camera for real-time ASL translation, providing a useful communication tool. Additionally, further improvements will be made by experimenting with different CNN architectures, adding more layers, and tuning hyperparameters for better performance. Lastly, expanding the dataset to include colored images and noise will increase the model’s robustness, enabling it to handle real-world conditions such as varying lighting and image noise.

In conclusion, this project demonstrates the potential of Convolutional Neural Networks (CNNs) for classifying American Sign Language gestures, with an emphasis on preprocessing and exploratory data analysis to prepare the dataset for model training. While the CNN provided promising results, challenges such as hyperparameter tuning and the impact of reduced image fidelity highlighted areas for improvement. Future work includes leveraging transfer learning, enhancing the model with live ASL translation, and expanding the dataset to increase robustness. Overall, the project serves as a foundation for further refinement and application of deep learning techniques in ASL recognition.