Machine Learning and Content Analytics Project: Sign Language Recognition:Video to text and speech

AUEB PT 22

Politis Kostas - p2822223

Spei Charalampia - p2822222

Tsatsi Giona - p2822226

CONTENTS

1.Introduction

Sign language has long been a cornerstone of communication for individuals with hearing loss. Many members of this group use it as a means of identification and primacy. Hearing impaired people are now able to communicate their needs, thoughts and feelings as effectively as those who can hear Due to the acquisition of sign language which is unique to many cultures and countries.

The walls that have traditionally divided communities are coming down as our society becomes more connected through technology. But despite the tremendous advances in communications technology, one glaring difference remains. For those who rely on sign language, everyday tasks—from attending an educational class to simply ordering a cup of coffee—can be difficult. This is not due to inability to communicate but to widespread ignorance of sign language. Many platforms, apps, and services aim for accessibility and fail to incorporate sign language recognition, resulting in haphazard communication.

This research is not limited to minor problems. The resulting impact is widespread. Misunderstandings can create a sense of exclusion for people with hearing loss who feel alienated from the community who do not 'speak' their language. Without visibly simple communication tools, those who use sign language as a primary means of communication may be excluded from certain professional and personal opportunities from certain employment opportunities or educational resources into sign language simply because most have not fully adopted or integrated the tools they use.

Thus, there's an imperative need for technology that bridges this communication divide, promoting a world where everyone, irrespective of their mode of communication, has equal access to opportunities and a sense of belonging.

1.1 Our Project

Our effort begins with the careful collection of video data, capturing a variety of human movements. These recordings are more than just canvases; They are the bedrock upon which our entire program is built. With this valuable data in hand, we use the versatile and powerful programming language Python in conjunction with the capabilities of the Keras library to create and train deep learning models Our neural network the types are varied. We use short-term memory (LSTM) networks, known for their proficiency in understanding sequences, and Gated Recurrent Units (GRU), a unique technique that provides a unique approach to visualization a the series looked In addition, we also examined two-way LSTM, that past It is also designed to enhance the contextual understanding of the model by considering future data

To prevent our models from overfitting and generalize well, we strategically detached the Dropout layers. But our commitment to a better model is not based on architectural design. We have included callback functions such as PlotLosses for real-time visual validation of the training process, providing a dynamic view of how our model is evolving. Once trained, the model is subjected to the prediction phase. Here, it has been the task to define craft, and to integrate these predictions against actual results to demonstrate the efficiency of the model.

Our research process is comprehensive. Using tools from the Scikit-learn library, we dive deeper into business data, breaking down confusion matrices to understand true positives, false negatives, and more. We also calculate accuracy scores to gain an overall understanding of the fit of the model.

Yet, the real marvel unfolds in real-time testing. We've ingeniously integrated Mediapipe's holistic model to detect human poses in live video. As actions are recognized, their probabilities get artistically overlaid on the video stream, and, in a symphony of technology, the recognized gestures are converted to spoken words using the pyttsx3 library. This capability doesn't just represent a technical achievement but stands as a testament to the endless possibilities of human-computer interaction.

1.2 Project Vision and Goals

At the core of our work is a profound vision: a world where communication itself is not a right, but a universally accessible opportunity, without traditional barriers or barriers. This is not only a high aspiration but a guiding principle that guides every aspect of our work. We see the technological advances of artificial intelligence as an unprecedented opportunity. Our goal is to carefully design an AI model that can interpret sign language in real time, ensuring that communication remains as spontaneous and realistic as a conversation between two individuals speaking in a channel which they live on.

The versatility of our AI model is one of its standouts. Rather than limiting what it can do, the system is designed to convert sign language into two separate but important components: audio and text. Such a theory ensures that whether one wants to hear an equivalent spoken symbol or read some of its text, both options are readily available, thus addressing a wide range of communication issues.

Contrary to some misconceptions, sign language is not a single or simple form of communication. It's a rich tapestry of intricate gestures, subtle expressions, and context-dependent adjustments that can be as detailed and complex as the spoken word and our A.I.

However, our vision extends beyond technology. It is deeply rooted in values of inclusion and equality. We are not just specimens; We build bridges—bridges that allow individuals with hearing loss to move freely through various aspects of their lives, whether it be academic spaces, professional corridors, or social gatherings, without feeling overwhelmed or overwhelmed by communication challenges they are limited.

Moreover, we are well aware of the variety of sign languages around the world. From American Sign Language to British Sign Language and beyond, each medium is a beautiful testament to the richness of human communication. While our immediate focus may be on one particular sign language, the underlying structure of our model is built to be scalable and scalable, holding the promise that future extensions will touch the world all of these varieties.

In our journey, we do more than just combine technological advances with human needs; We form a harmonious alliance between them. Our goal, albeit ambitious, clearly remains the same: to create a system of tomorrow where every connection is ubiquitous and infinite, and to connect human networks in ways we have yet to imagine.

2.Methodology

2.1 Data Collection

In the comprehensive endeavor to architect an intricate gesture recognition system, our team, comprising three dedicated members, painstakingly engaged in a rigorous process of data collection. Recognizing the pivotal role that environment plays in acquiring clear and consistent data, we selected a setting with stable lighting and ensured an absence of potential distractions or disturbances. The essence of our data revolved around five pre-defined phrases: 'hello', 'I love you', 'will you', 'tell me', and 'your name'. Each phrase corresponded to a distinct gesture, and every member was responsible for enacting these gestures in front of a high-definition camera. We consciously embraced our individual nuances and variations in our gesture execution, which was paramount in diversifying our dataset. With each of us bringing a unique flair to the gestures, we were able to collate a rich repository of data, capturing subtle intricacies and nuances. This vast pool of information wasn't just a chronological sequence of hand movements; it was a compendium of our individualities and distinct interpretations of each phrase. Such a holistic and comprehensive dataset is primed to be the foundation upon which a robust and versatile deep learning model can be trained, ensuring its reliability and wide applicability in real-world scenarios.

2.2 Data Overview

Our project's foundation is built upon the formidable capabilities of Google's mediapipe library, known for its expertise in the realm of computer vision. This library is at the heart of our gesture mapping and analysis.

During the initial phase, the solution interfaces with a device's primary camera in real-time using the OpenCV library, often referred to by its abbreviation, cv2. This real-time capture is critical to ensure that the gestures, whether rapid or slow, are captured in their entirety. After capturing, every video frame goes through a vital transformation where its color space is converted from BGR to RGB. This step, although seemingly technical and mundane, is pivotal. The ***mediapipe*** holistic model, in which we extensively rely on, requires its input frames to be in the RGB spectrum. This

conversion ensures that our data is in the right format, primed for accurate gesture detection and analysis.

Progressing deeper into our methodological framework, we arrive at a juncture where the ***holistic.process*** function plays a quintessential role. This function, with its robust algorithms, is responsible for identifying a comprehensive set of holistic body landmarks. These aren't your ordinary set of data points. They capture the very essence of human expressiveness – from the subtle raise of an eyebrow to the broad motion of an arm swing.

Now, while capturing these landmarks is an achievement in itself, the real challenge lies in visualizing them in a manner that's insightful. That's where our two-pronged visualization strategy steps in. The first approach, termed as draw_landmarks, serves as our foundational method. At its core, it's engineered to effectively plot and showcase the detected landmarks on the source imagery.

Also ,entering function ***draw_styled_landmarks***. This isn't just another visualization tool. It's a culmination of our insights into human gesture interpretation, backed by the power of customization. Unlike its more generic counterpart, this method is equipped to add flair and differentiation to the visual output. With it, we can assign distinct visualization styles to different landmark sets. Imagine being able to visually distinguish a hand gesture from a facial cue using different color codes, line types, or marker styles. That's the power ***draw_styled_landmarks*** brings to the table. The result? A rich, layered, and intuitive representation that not only highlights gestures but also tells a vivid story of their interplay and significance.

The crux of our methodology isn't just about mere detection but in comprehensively understanding and analyzing the rich data we extract. It is in this phase of intricate data analysis that the roles of the functions ***extract_keypoints*** and ***calculate_features*** become undeniably pivotal.

Upon the successful detection of the holistic landmarks, we are presented with a myriad of data points. At a superficial glance, these might appear as mere scattered

reference points. However, beneath this facade, they hold a wealth of information waiting to be unveiled. Rather than leaving them in their raw state, we employ the ***extract_keypoints*** function, transforming these scattered landmarks into a structured data format. What we obtain post this transformation is an ensemble of coordinates. Each landmark yields its x, y, and z positions, crafting a three-dimensional representation of the gesture. Furthermore, intertwined within these coordinates are the visibility scores, which shed light on the prominence and relevance of each landmark in the captured frame.

Our feature set, the outcome of this extraction, is both diverse and profound. It encapsulates a multitude of aspects from the detected frame. From the pose landmarks that map the overall stance and posture of the user to the facial landmarks that might hint at facial expressions or nuances, we capture it all. However, the depth doesn't end there. We also focus precisely on the finer points of hand movements, describing the specifics of both the left and right hands. Each finger, each joint, and its relative positioning is mapped, ensuring no gesture, no matter how nuanced, escapes our analysis.

Yet, the simple extraction of these landmarks isn't the zenith of our process. With the calculate_features function, we delve even deeper. Gestures, in essence, are about relationships – the distance between the thumb and the index finger, the angle formed by the wrist's tilt, and myriad other spatial relationships that give meaning to a motion. By meticulously calculating the distances between landmarks and decoding the angles they form with one another, we're doing more than just data analysis; we're interpreting. This intricate process of understanding spatial relationships and patterns is our key to unraveling the essence of gestures, allowing us to perceive beyond mere motions and grasp the underlying intent.

Moreover, it's essential to recognize that our data is sequenced with precision. It's structured to be action-specific, aligning with our target gestures such as 'hello', 'I love you', and the like. For data collection, the code mandates capturing 90 video sequences (or no_sequences) for each of these actions. Each of these sequences, in turn, is about 30 frames in length (as per the sequence_length parameter),

approximating a second-long video at standard frame rates. Such a setup ensures the system captures a robust variety of samples for each gesture, laying a strong foundation for a high-accuracy model.

But what truly elevates our data collection methodology is our focus on realism. We haven't limited ourselves to perfect scenarios. By adjusting brightness and contrast of frames, we simulate a myriad of lighting conditions. This robust approach ensures that our dataset is adept at handling real-world challenges. Plus, by storing mirrored versions of frames, we're enhancing dataset diversity, making sure our eventual model remains unbiased and versatile.

Conclusively, the final resting place for this data, after all these meticulous processes, is the specifically designated storage path.Here, the processed data will be systematically arranged into action-specific directories and further segmented into subdirectories reflecting individual sequences. The sheer complexity of our data collection combined with the depth of features we extract confidently positions us on the forefront to create a gesture recognition model that is not only accurate but also captures the finer nuances of human gestures.

2.3 Data Processing/Annotation/Normalization

Our gesture recognition framework is designed to encompass a series of comprehensive steps starting from raw data ingestion to its refinement and preparation for model training. The focus of this report is to provide a deep dive into the intricacies of this process.

The initial stage, Data Loading and Annotation, is crucial to establish the foundation for subsequent operations. Here, the code leverages the predefined list of actions, which essentially contains labels of the gestures. Each action or gesture in this list is mapped to a unique integer identifier using the label_map dictionary, a step pivotal for data annotation and eventual model training. With this mapping in place, our system traverses through the dataset, sequentially reading each frame for every action and its corresponding sequence. As each frame is ingested, it is stacked into a 'window' or sequence, which once complete, is appended to a larger dataset denoted

as sequences. Simultaneously, the associated action's integer label, derived from the label_map, is added to the labels list.

Following data ingestion, the Data Splitting phase takes precedence. The partitioning of the data for model training, validation, and testing is ensured by this stage. The dataset is initially split into a 20% temporary subset (X_temp and y_temp) and an 80% training subset (X_train and y_train) using scikit-learn's train_test_split function. This temporary set is divided again into equal halves for testing (X_test and y_test) and validation (X_val and y_val). Such partitioning is imperative to avoid data leakage and to evaluate the model's performance on unseen data effectively.

The final step of the journey, Data Augmentation, is centered on enhancing the model's generalization capabilities. Two specific augmentation techniques are incorporated: temporal translation and jittering of keypoints. Temporal translation is essentially a shift in the video frames, achieved using the np.roll function, creating a slight temporal displacement in the sequence. This is especially valuable in scenarios where gestures might start a tad earlier or later in the video. Jittering, on the other hand, introduces minor noise to the keypoints in the video, simulating potential real-world discrepancies in landmark detection. Both augmentations lead to a more diverse training set, making the model robust against variations.

Each augmentation process follows a similar pattern. For every video in the training set, an augmented version is generated and stored in an interim list (X_augmented and y_augmented). Once all videos are processed, these augmented datasets are converted to numpy arrays and amalgamated with the original training dataset, effectively doubling its size.

Conclusively, the code embodies a structured and meticulous approach to data processing, annotation, and normalization. From accurately labeling gestures to splitting the dataset and introducing meaningful augmentations, each step is crafted to ensure that the ensuing gesture recognition model is trained on a diverse, comprehensive, and well-structured dataset.

2.4. Algorithms, NLP architectures/system

2.4.1Building and Training the LSTM Neural Network

In the intricate realm of machine learning, the code presented gives insights into the deployment of state-of-the-art neural network architectures for specific tasks. This code hinges on the power of the Keras library from TensorFlow. Keras, renowned for its high-level neural networks API, offers the flexibility of a Python-based interface and compatibility with backend systems like TensorFlow. A distinguishing feature of this library is the Sequential model. With its capacity for a linear stack of layers, it finds its perfect application in scenarios demanding a single input and output for the model, aligning seamlessly with this project's demands.

Diving deeper into the code, there's an apparent exploration of various neural network architectures, particularly Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRU), and their bidirectional counterparts. LSTMs and GRUs, both variants of Recurrent Neural Networks (RNNs), are adept at handling sequences, making them invaluable for tasks like gesture recognition where temporal relationships play a crucial role. The choice between them often comes down to the trade-off between computational efficiency and modeling power.Initially, dependencies from the Keras library were imported. These dependencies serve as the building blocks of our models, facilitating seamless neural network design and execution. Among these is the unique PlotLosses class, an integral tool to visualize our model's training progress. As epochs progress, this callback displays real-time loss and accuracy graphs, providing a tangible insight into how well the model is adapting to the data.

The crux of this phase involves architecting and experimenting with different neural network structures:

LSTM with ReLU Activation: This architecture involves three LSTM layers with the Rectified Linear Unit (ReLU) activation function. The "return sequences" parameter ensures that every LSTM unit returns an output, thus maintaining the sequence's integrity. Furthermore, Dropout layers are interspersed between LSTMs and Dense layers to prevent overfitting.

GRU with ReLU Activation: A variant of the RNN, the GRU model was tested with a similar three-layer structure using the ReLU activation function. GRUs are often considered more efficient than LSTMs for certain tasks, as they have fewer parameters.

LSTM with Sigmoid Activation: Another variant was tested using LSTMs but with a Sigmoid activation function. The Sigmoid function constrains the output between 0 and 1, making it particularly suited for binary classification problems.

Bidirectional LSTM with Sigmoid Activation: The Bidirectional LSTM wraps around standard LSTM layers, ensuring that the input sequence is processed in both forward and backward directions. This can offer a deeper understanding of the context, often leading to better performance on certain tasks.

GRU with Sigmoid Activation: A model configuration similar to the ReLU-based GRU, but this time leveraging the Sigmoid activation function.

For each of these model architectures, Dense layers were added at the end, concluding with a softmax activation function which is a typical choice for multi-class classification problems.

In the initial phases of developing the profound learning model, different structural and adjustment techniques were contemplated. After defining the architecture, the team investigated several adjustment techniques, aside from the widely preferred Adam optimizer. Even though other optimizers showed some potential, the exceptional amalgamation of the momentum and adaptive learning rates in Adam made it standout in relation to convergence speed and overall performance on the validation set.

Post architecture definition, the final model was compiled using the Adam optimizer, which is renowned for its effectiveness in deep learning scenarios. The categorical cross-entropy loss function, being a standard choice for multi-class problems, was utilized. For monitoring performance, categorical accuracy was chosen as the key metric. To further enhance training, an early stopping callback was used, which

intelligently halts training if no significant improvement is observed, preventing unnecessary computational expenditure.

Finally, the model was trained on the training dataset (X_train and y_train). The training phase was set for a generous 2000 epochs, although early stopping could terminate it sooner based on the model's performance.

Upon training completion, the model summary illustrated a meticulous breakdown of the network layers, output shapes, and parameter counts, providing a comprehensive view of the underlying architecture and its complexity.

In conclusion, this deep dive into recurrent neural networks, through varied architectures and configurations, offers a robust foundation to harness the temporal nature of gesture data. By understanding sequences in depth, it paves the way for creating a potent gesture recognition system, adept at capturing the nuances and intricacies of human movements.

2.4.2 Making Predictions

Following the meticulous training process, the next critical step in the machine learning pipeline is to gauge the model's performance on unseen data, typically encapsulated in the prediction phase. In this code segment, the model's prowess is put to the test against both the 'test' and 'validation' datasets.

Starting with the test dataset, predictions are made using the predict method of the model. By using the numpy argmax function, the index of the highest probability in the output array is identified, which corresponds to the predicted class. It's subsequently mapped back to the original 'actions' array to retrieve the human-readable label. For the specific instance shown in the code, the model successfully identifies the gesture as "will you", aligning perfectly with the ground truth.

Similarly, the validation data undergoes a prediction exercise. The emphasis on validating performance on multiple data sets underscores the importance of ensuring model robustness across different data distributions. The sample from the validation dataset presented yields the prediction "your name". Once again, the model's prediction mirrors the actual label, indicating the model's commendable accuracy.

It's important to note that while the highlighted predictions are accurate, an exhaustive assessment would involve evaluating the model's performance across the entire test and validation datasets. Typically, metrics such as accuracy, F1 score, precision, and recall provide a comprehensive picture of the model's capabilities.

2.4.3 Save and Load Weights

Post-training, one of the paramount stages in a machine learning pipeline is preserving the model's state. This includes the architecture, weights, biases, and even the optimizer's state, to facilitate model reuse, deployment, or further fine-tuning in the future. In this segment of the code, the focus is on the management of the model's weights, which are the tuned parameters acquired during the training phase.

Initially, the model's state is saved to a file named 'GRU.h5' using the save method. The '.h5' file format signifies the usage of HDF5 (Hierarchical Data Format version 5), a popular file format for storing large datasets, especially arrays of numbers. This format is especially suited for efficiently storing model weights due to its support for quick read and write operations.

Following the preservation of the model, it is intentionally deleted from the memory with the del statement, simulating a scenario where the initial training environment is no longer available or the context has changed. Such actions highlight the importance of saving model states, as recreating a model from scratch or retraining it would be resource-intensive and time-consuming.

To demonstrate the efficacy of the saving process, an attempt is then made to load the weights from a file named 'LSTM.h5' into the model using the load_weights method. It's noteworthy that while the saved file was named 'GRU.h5', the loaded file is 'LSTM.h5'. This suggests that there might be multiple models at play, and it's imperative to ensure compatibility between the architecture used during saving and loading. The weight-loading process is instrumental in scenarios like transfer learning or when you need to swiftly switch between different model states.

In sum, the process of saving and loading model weights is quintessential for the continuity and scalability of machine learning projects. It not only ensures that the time and resources spent during training are not in vain but also provides flexibility in deployment and further enhancements of the model. The 'GRU.h5' and 'LSTM.h5' files now serve as checkpoints of the model's progress and can be instrumental in various future applications.

2.4.4 Evaluation using Confusion Matrix and Accuracy

In the evaluative phase of our neural network, a dedicated section is orchestrated to scrutinize its performance, primarily via confusion matrices and accuracy metrics, both powerful tools in the domain of classification problems. The onset of this evaluation witnesses the importation of pertinent functions — multilabel_confusion_matrix and accuracy_score — from the well-regarded sklearn.metrics module. Their inclusion signifies the preparatory step for a meticulous examination of the model's prowess.

Before any performance assessment can occur, predictions need to be procured. As such, the predict method of the model is applied to X_test, resulting in yhat, which essentially is a matrix of predicted probabilities. It's crucial to understand that each entry in this matrix showcases the model's confidence regarding a particular class assignment. To facilitate a comparison between predicted and true labels, both are transformed from their current matrix forms to a more digestible list of class indices using np.argmax.

The multilabel confusion matrix is then computed, serving as a profound diagnostic tool. For the uninitiated, each matrix within this larger multilabel matrix details the performance for a particular class, indicating true positives, false positives, true negatives, and false negatives. It's an essential tool for multiclass problems, offering insights that aren't just limited to global performance but extend to class-specific dynamics, shedding light on potential areas where the model might be faltering.

In tandem with the confusion matrix, the accuracy score is gleaned. While the confusion matrix is granular, the accuracy score provides a bird's-eye view of the

model's performance, reflecting the proportion of correct predictions. For the test dataset, the model secured an accuracy of 0.2, which, when interpreted, suggests that it correctly predicted 20% of the test cases.

Subsequently, a similar methodology is adopted for the validation dataset (X_val). This repeated assessment is pivotal, ensuring that the model's performance is consistent and not limited to just the test dataset. However, a comparison of results reveals an evident disparity. The validation dataset clocked an accuracy of roughly 0.067, considerably lower than that of the test set. Such a divergence in performance is an invitation to delve deeper into potential areas of improvement, be it refining the model's architecture, reconsidering training techniques, or reevaluating preprocessing measures.

3.Experiments & Results

3.1 Experiments – Setup, Configuration

In the domain of our experiments, the live testing phase stands out as a testament to the system's effectiveness and resilience. We have diligently constructed a setup that not only detects actions but also provides immediate feedback, making the user experience both interactive and informative. The initial step is the establishment of a vibrant color scheme through the colors array. These shades serve a dual purpose. They are not just for aesthetics but are functionally crucial in distinguishing between the various actions in real-time. This visual differentiation is made possible by the prob_viz function.

Upon receiving the model's predicted actions, alongside the current video frame and the set colors, this function overlays the frame with colored rectangles. The width of these rectangles is indicative of the probability associated with each action, offering a clear visual cue about the system's confidence in its predictions. Simultaneous feedback isn't just visual. Utilizing the pyttsx3 library, the system audibly announces the recognized actions, thereby ensuring that feedback is not just seen but also heard. This audio output is managed by text_to_speech, which converts text into speech. To prevent any delays or interruptions, convert_to_speech_in_thread initiates this audio feedback in a separate thread, ensuring smooth execution.

For the real-time discovery, several variables are initialized. The order captures the order of keypoints, phrase logs the activities, and identified_terms keeps track of spoken words. A unique characteristic is the introduction of a threshold. Only when the model's certainty surpasses this threshold does the system acknowledge the activity, ensuring that feedback is based on relatively certain predictions.

Using OpenCV, the system taps into the webcam feed. As frames stream in, the mediapipe model, represented by mp_holistic.Holistic, detects and tracks bodily landmarks. These landmarks are then stylized and rendered onto the frame. But more importantly, they are also processed into keypoints, which feed into our trained model to predict activities. As activities are predicted, they are audibly announced. The visual feedback is twofold: firstly, through the dynamic colored rectangles representing

activity probabilities and secondly, through a textual display of the most recent activities and the model's certainty in its latest prediction.

This combined feedback ensures users are always in the loop regarding the system's perceptions. Finally, the system doesn't just operate in a vacuum. A user can gracefully end the real-time testing by pressing the 'q' key, post which the identified activities are collated into a text and printed.

In summary, the real-time testing setup is a confluence of sophisticated techniques and user-centered design, promising an experience that's as enlightening as it is engaging.

3.2 Advanced Speech Synthesis using Transformer-based Models

Following the real-time action detection and vocal announcements, our experiment introduces another intriguing dimension - advanced speech synthesis. This capability underscores the system's potential in transforming raw data into intelligible and nuanced audible feedback.

To embark on this venture, we harness the power of the transformers library, which is a reservoir of pre-trained models specializing in Natural Language Processing tasks. Notably, we exploit the AutoProcessor and AutoModel classes. These classes are instrumental in adapting and fine-tuning pre-trained models for specific tasks, in this case, speech synthesis.

Our chosen model, denoted as "suno/bark-small," is initialized via these classes. This model, pre-trained on an expansive dataset, comes equipped to convert textual information into a speech format. It's worth noting that the name "suno/bark-small" could be indicative of a lightweight model optimized for swift processing, although maintaining a respectable level of audio fidelity.

The synthesis process is fairly straightforward. The recognized words, collated into recognized_text, are first fed into the processor. This processor, tailored to work seamlessly with our chosen model, converts the raw text into tensors. Tensors, essentially multi-dimensional arrays, are the primary data structure that deep learning

models operate on. Once the text is tensorized, it's processed by the model, resulting in speech values. These values, when played at the appropriate sampling_rate, reconstruct the audible speech.

To ensure the synthesized speech mirrors human tonality and inflection, the model utilizes the do_sample=True parameter. This introduces a degree of randomness, ensuring that the generated speech isn't monotonous but has a more organic sound to it.

The IPython.display.Audio utility then plays this generated speech, enabling users to audibly discern the content of recognized_text.

In conclusion, this segment of the experiment accentuates the synergy between computer vision and advanced NLP techniques. By converting visually recognized actions into intricate speech, the system exemplifies the potential of multi-disciplinary AI applications. Not only does this cater to diverse user needs, but it also paves the way for more integrated and holistic AI-powered solutions in the future.

3.3 Comparison of Different Neural Network Models

In our quest to identify the most effective neural network model for our dataset, we examined the performance of various architectures, including LSTM with relu activation, GRU with relu activation, LSTM with sigmoid activation, Bidirectional LSTM with sigmoid activation, and GRU with sigmoid activation. The evaluation criteria were based on precision, recall, f1-score, and overall accuracy on both test and validation datasets.

3.3.1 LSTM relu Performance:

The LSTM relu model's performance left much to be desired. On the test dataset, it yielded an accuracy of only 18%, while on the validation dataset, it underperformed even more with an accuracy of 13%. The model's proficiency in recognizing actions from the given dataset appears to be lacking. While it does make predictions, its overall success rate is limited, often missing the mark. Conversely, actions like "hello", "i love you", and "tell me" are consistently overlooked by the model, indicating

significant areas of improvement. The combined measure of precision and recall, the F1-scores, highlights the model's challenges. In its current form, the model falls short of providing a reliable and consistent performance for this specific action recognition task (Figures 1 to 4).

3.3.2 GRU relu Performance:

The GRU relu model showed a marked improvement over the LSTM relu. It achieved an 80% accuracy on the test dataset and an impressive 87% on the validation dataset. The model's performance is notably impressive, showcasing a marked capability in recognizing actions. Looking at the confusion matrix, we observe strong diagonal values, which suggests a high level of correct predictions across the various classes. The "hello" class, for instance, seems to have been predicted with both confidence and precision. Similarly, classes like "will you" and "tell me" exhibit commendable results. While the "i love you" class had some challenges, the model still managed to provide valuable predictions. The classification report further emphasizes this model's robustness, with many classes achieving a balance between precision and recall. On the whole, this model demonstrates a solid proficiency in action recognition, making reliable predictions across different categories (Figures 5 to 8).

3.3.3 LSTM SIGMOID Performance:

The LSTM model with a sigmoid activation function has delivered an exceptionally robust performance on the given dataset. With an impressive accuracy of approximately 98%, the model is adept at correctly identifying almost all test samples. Most of the classes, such as "hello", "i love you", and "will you", have been recognized with impeccable precision and recall, indicating both the model's confidence in its predictions and its ability to capture all instances of these actions. The class "tell me" has a remarkable precision, though its recall is slightly reduced, implying that the model missed some instances of this action. For the "your name" class, while the precision is slightly lower, the perfect recall indicates that the model successfully identifies all instances of this action, even if it occasionally misclassifies other actions under this label. When considering the F1-scores, which provide a balanced view of

precision and recall, the model's capabilities are further emphasized. All these metrics highlight the LSTM sigmoid model's outstanding competency in recognizing actions for the specified dataset (Figures 9 to 12).

3.3.4 Bidirectional SIGMOID Performance:

The standout performer among the models evaluated was the Bidirectional LSTM with sigmoid activation. This model achieved perfection with an accuracy of 100% on both the test and validation datasets. Delving deeper into individual classes such as "hello", "i love you", "will you", "tell me", and "your name", both precision and recall are spot-on at 100% for each class. This means not only did the model correctly identify every instance of each action, but it also made no false predictions. The F1-scores, which provide a harmonic mean of precision and recall, further reinforce this perfection by being set at 1.00 across all classes. Such results are indicative of a model that has been finely tuned and is highly adept at action recognition for this specific context.

While this is a commendable feat, caution is advised. A perfect score might indicate a possibility of overfitting. It's crucial to validate this model further on unseen data to ensure its generalization capabilities (Figures 13 to 16).

3.3.5 GRU SIGMOID Performance:

The GRU model with sigmoid activation was another top contender. It nearly mirrored the performance of the Bidirectional LSTM model on the validation set with a perfect score. With an overarching accuracy of 98%, the model's proficiency is evident. Actions such as "hello", "I love you", and "your name" have been flawlessly identified, with perfect scores in precision, recall, and F1-score. The action "will you", although showing a slightly reduced precision of 88%, compensates with an impeccable recall, indicating that while the model occasionally mislabels other actions as "will you", it never misses an actual instance of this action. The "tell me" class, while pinpoint accurate in its predictions, misses the mark slightly in recall, indicating that there are rare instances where the model doesn't identify this action. Across the board, the averaged metrics are consistently high, hovering around the 97-98% mark. Such performance speaks volumes about the model's adeptness, although it's crucial to

remain vigilant about potential overfitting and to ensure the model's robustness in diverse settings (Figures 17 to 20).

3.3.6 Conclusion

Upon comparing the Bidirectional LSTM and GRU models, it becomes evident that the GRU model is more suitable for our needs. This decision is primarily based on its superior accuracy. Furthermore, despite the impressive performance of the Bidirectional LSTM, there's a palpable concern regarding its potential to overfit, which further cements our preference for the GRU model in practical applications. This conclusion is even more evident when we try the two models on the live testing phase.

3.4 Qualitative & Error Analysis

The process of analyzing transcription errors in ASR systems is multifaceted. The primary step involves a qualitative analysis, where patterns in misinterpretations, like consistent mistakes with specific words or the impact of background noise, come to light. This analysis feeds into a more structured error categorization, breaking down mistakes into types: substitution errors, where words are replaced; deletion errors, indicating omitted words; and insertion errors, noting unnecessary additions. To refine the system's accuracy, several strategies are employed. The system could be retrained using diverse data, ensuring a broader understanding of speech nuances. Additionally, post-processing steps can be introduced, making corrections after the initial transcription. An optimal recording environment, free from disruptions, also plays a pivotal role in ensuring accuracy. User feedback, a crucial component, not only aids in identifying discrepancies but also fosters user trust and engagement. If challenges persist, pivoting to alternative ASR systems, better suited to specific linguistic nuances, might be the way forward.

3.4.1 Qualitative Analysis

During the evaluation phase, we delved into the experiential aspect of our model's performance, specifically in the context of real-time camera-based word recognition. This qualitative approach allowed us to appreciate and better understand the nuances of the system's real-world applicability. The most noteworthy observation was the system's temporal responsiveness. At certain intervals, the camera exhibited a

perceptible delay in recognizing words. This lag not only impeded the seamless transition of recognition tasks but also affected the user experience, highlighting an area of potential improvement.

Moreover,while our model successfully translated the list of signs into audible speech, the proceudre wasn't without its challenges. The final audio output, although correct in content, often sounded artificial and lacked the fluidity and warmth of natural human speech. Utilizing state-of-the-art tools like the "suno/bark-small" model from the transformers library, we anticipated a lifelike vocal rendition of the signs. However, the result, at times, felt more robotic than human. This underscores a broader challenge in the field of speech synthesis: replicating the nuanced intonations, pauses, and rhythms that make human speech unique and engaging. As we move forward, enhancing the naturalness of the audio output will be a priority, ensuring that our system not only communicates the right words but does so in a way that resonates authentically with the listener.

3.4.2 Error Analysis

A deeper investigation into the aforementioned delay brought forth a set of challenges that our model encountered. Primarily, overlapping was a recurrent issue. Instead of cleanly transitioning from one word to the next, the system occasionally either overlapped with the previous word or prematurely progressed to the succeeding word. This overlapping phenomenon, while sporadic, poses a significant hurdle. It introduces ambiguity and reduces the precision of word recognition, which is paramount for applications necessitating accurate and swift interpretation.

3.4.3 Challenges Faced During Model Execution

The challenges that emerged were not strictly limited to post-deployment performance. During the model's execution, the inherent complexity of real-time word recognition manifested in unexpected ways. The intricacies of capturing words in varying environmental conditions, differences in pronunciation speed, and even slight deviations in word articulation may have contributed to the aforementioned camera delays and overlapping issues. Addressing these issues would require a multifaceted approach, combining model refinement, enhanced training datasets,

and potentially even hardware optimizations to ensure timely and accurate recognition.

5. Discussion, Comments/Notes and Future Work

In the pursuit of refining our sign language recognition system, several avenues for future enhancements emerge. A foundational step would be to enrich our dataset by capturing more diverse samples that cater to varied lighting conditions, backgrounds, and a wider demographic spectrum. Advanced data augmentation techniques, extending beyond the temporal translation and jittering used in this study, could further diversify the training set and bolster the model's generalization capabilities. Experimentation with alternative neural architectures, particularly those like Transformers, could provide performance boosts. Regularization techniques and systematic hyperparameter tuning might further optimize the learning process. The potential of transfer learning, leveraging pre-trained models on analogous tasks and adapting them to our dataset, remains an enticing proposition.

A more nuanced evaluation of our model can be achieved by broadening our metric suite, possibly incorporating measures that scrutinize model calibration. Given the temporal nature of sign language, post-processing techniques can be explored to ensure the continuity and coherence of recognized gestures. A context-aware model, which considers preceding signs to inform current predictions, could amplify accuracy in real-world scenarios. Transitioning from a purely technical perspective, user-centric improvements are paramount. An interactive, intuitive interface can significantly elevate the user experience. By expanding the model's vocabulary and integrating a real-time feedback mechanism, we can render the system both versatile and educational. Lastly, a focus on model interpretability can shed light on the model's decision-making process, promoting transparency and facilitating error diagnosis.

6. Members/Roles

Study/Research: Spei,Politis,Tsatsi

Data Collection: Spei, Politis

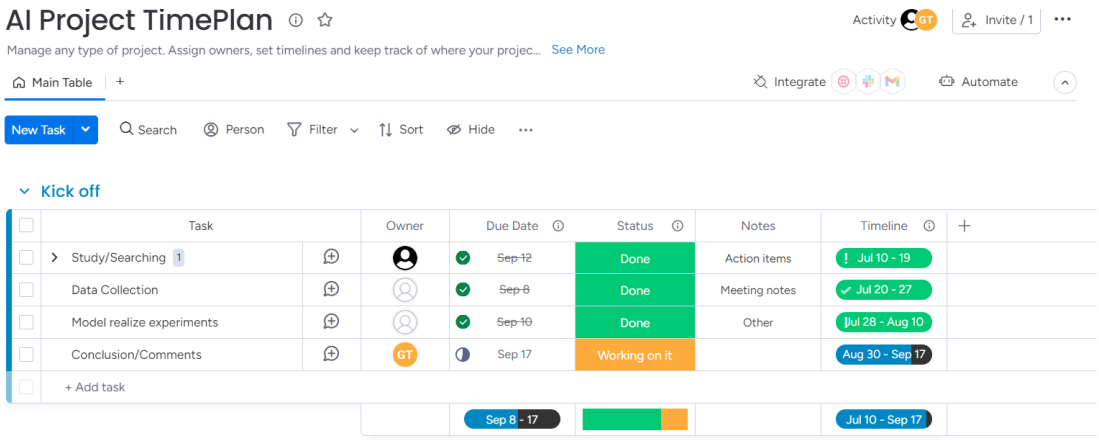
Model & algorithms: Spei, Politis

Text to audio:Spei,Tsatsi

Realizing experiments: Tsatsi,Politis

Report/Presentation: Spei,Tsatsi,Politis

7. Time Plan



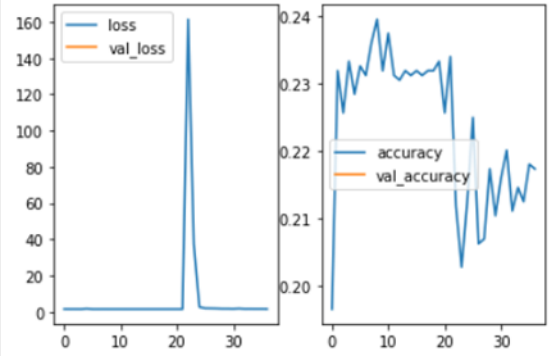
8.Bibliography

Aggarwal, C.C. (2023) *Neural networks and deep learning: A textbook*. Cham, Switzerland: Springer.

Géron, A. (2023) *Hands-on machine learning with scikit-learn, keras and tensorflow: Concepts, tools, and techniques to build Intelligent Systems*. Sebastapol, CA: O'Reilly.

9. Appendix

LSTM relu




```
45/45 [==============================] - 4s 80ms/step - loss: 1.6597 - categorical_accuracy: 0.2174
```

Figure 1:LSTM relu Plot Loss

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_12 (LSTM)              (None, 30, 64)            453632

 dropout_25 (Dropout)        (None, 30, 64)            0

 lstm_13 (LSTM)              (None, 30, 128)           98816

 dropout_26 (Dropout)        (None, 30, 128)           0

 lstm_14 (LSTM)              (None, 64)                49408

 dropout_27 (Dropout)        (None, 64)                0

 dense_15 (Dense)            (None, 64)                4160

 dropout_28 (Dropout)        (None, 64)                0

 dense_16 (Dense)            (None, 32)                2080

 dropout_29 (Dropout)        (None, 32)                0

 dense_17 (Dense)            (None, 5)                 165

=================================================================
Total params: 608,261
Trainable params: 608,261
Non-trainable params: 0
```

Figure 2:LSTM relu Model Summary

```
2/2 [==============================] - 0s 20ms/step
Multilabel Confusion Matrix:
[[[33  0]
  [12  0]]

 [[36  0]
  [ 9  0]]

 [[ 1 37]
  [ 0  7]]

 [[38  0]
  [ 7  0]]

 [[35  0]
  [ 9  1]]]

Accuracy Score: 0.17777777777777778

Classification Report:
              precision    recall  f1-score   support

       hello       0.00      0.00      0.00        12
  i love you       0.00      0.00      0.00         9
    will you       0.16      1.00      0.27         7
     tell me       0.00      0.00      0.00         7
   your name       1.00      0.10      0.18        10

    accuracy                           0.18        45
   macro avg       0.23      0.22      0.09        45
weighted avg       0.25      0.18      0.08        45
```

Figure 3:LSTM relu Confusion Matrix/ Metrics Test Data

```
2/2 [==============================] - 0s 23ms/step
Multilabel Confusion Matrix for Validation Set:
[[[34  0]
  [11  0]]

 [[34  0]
  [11  0]]

 [[ 0 39]
  [ 0  6]]

 [[35  0]
  [10  0]]

 [[38  0]
  [ 7  0]]]

Accuracy Score for Validation Set: 0.13333333333333333

Classification Report for Validation Set:
              precision    recall  f1-score   support

       hello       0.00      0.00      0.00        11
  i love you       0.00      0.00      0.00        11
    will you       0.13      1.00      0.24         6
     tell me       0.00      0.00      0.00        10
   your name       0.00      0.00      0.00         7

    accuracy                           0.13        45
   macro avg       0.03      0.20      0.05        45
weighted avg       0.02      0.13      0.03        45
```

Figure 4: LSTM relu Confusion Matrix/ Metrics Validation Data

GRU relu

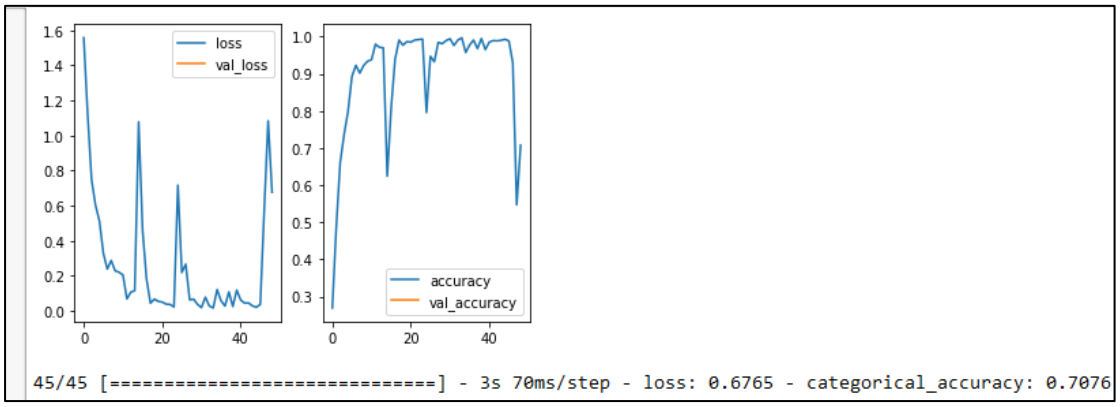


Figure 5: GRU relu Plot Loss

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_3 (GRU)                 (None, 30, 64)            340416

 dropout_30 (Dropout)        (None, 30, 64)            0

 gru_4 (GRU)                 (None, 30, 128)           74496

 dropout_31 (Dropout)        (None, 30, 128)           0

 gru_5 (GRU)                 (None, 64)                37248

 dropout_32 (Dropout)        (None, 64)                0

 dense_18 (Dense)            (None, 64)                4160

 dropout_33 (Dropout)        (None, 64)                0

 dense_19 (Dense)            (None, 32)                2080

 dropout_34 (Dropout)        (None, 32)                0

 dense_20 (Dense)            (None, 5)                 165

=================================================================
Total params: 458,565
Trainable params: 458,565
Non-trainable params: 0
```

Figure 6: GRU relu Model Summary

```
2/2 [==============================] - 0s 14ms/step
Multilabel Confusion Matrix:
[[[32  1]
  [ 0 12]]

 [[34  2]
  [ 5  4]]

 [[33  5]
  [ 2  5]]

 [[37  1]
  [ 1  6]]

 [[35  0]
  [ 1  9]]]

Accuracy Score: 0.8

Classification Report:
              precision    recall  f1-score   support

       hello       0.92      1.00      0.96        12
  i love you       0.67      0.44      0.53         9
    will you       0.50      0.71      0.59         7
     tell me       0.86      0.86      0.86         7
   your name       1.00      0.90      0.95        10

    accuracy                           0.80        45
   macro avg       0.79      0.78      0.78        45
weighted avg       0.81      0.80      0.80        45
```

Figure 7: GRU relu Confusion Matrix/ Metrics Test Data

```
2/2 [==============================] - 0s 14ms/step
Multilabel Confusion Matrix for Validation Set:
[[[34  0]
  [ 2  9]]

 [[34  0]
  [ 3  8]]

 [[34  5]
  [ 0  6]]

 [[34  1]
  [ 0 10]]

 [[38  0]
  [ 1  6]]]

Accuracy Score for Validation Set: 0.8666666666666667

Classification Report for Validation Set:
              precision    recall  f1-score   support

       hello       1.00      0.82      0.90        11
  i love you       1.00      0.73      0.84        11
    will you       0.55      1.00      0.71         6
     tell me       0.91      1.00      0.95        10
   your name       1.00      0.86      0.92         7

    accuracy                           0.87        45
   macro avg       0.89      0.88      0.86        45
weighted avg       0.92      0.87      0.88        45
```

Figure 8: GRU relu Confusion Matrix/ Metrics Validation Data

LSTM SIGMOID

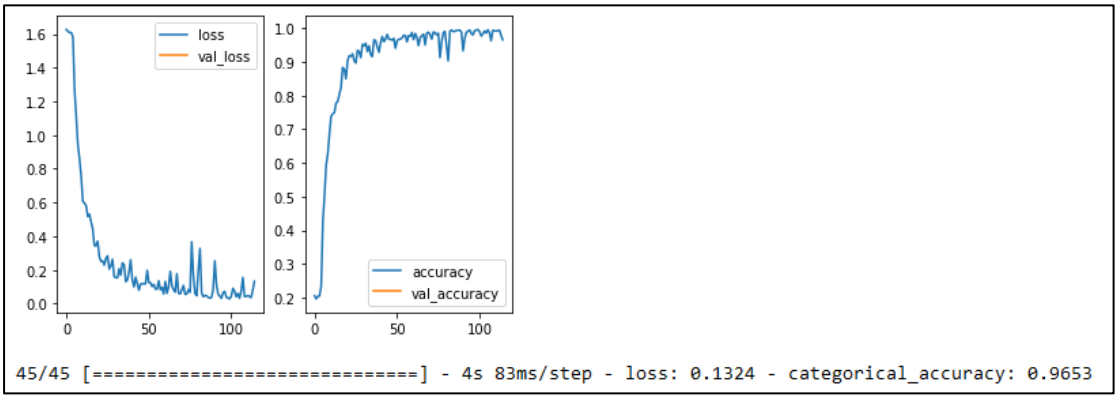


Figure 9: LSTM SIGMOID Plot Loss

Layer (type)	Output Shape	Param #
lstm_21 (LSTM)	(None, 30, 64)	453632
dropout_45 (Dropout)	(None, 30, 64)	0
lstm_22 (LSTM)	(None, 30, 128)	98816
dropout_46 (Dropout)	(None, 30, 128)	0
lstm_23 (LSTM)	(None, 64)	49408
dropout_47 (Dropout)	(None, 64)	0
dense_27 (Dense)	(None, 64)	4160
dropout_48 (Dropout)	(None, 64)	0
dense_28 (Dense)	(None, 32)	2080
dropout_49 (Dropout)	(None, 32)	0
dense_29 (Dense)	(None, 5)	165

Total params: 608,261
Trainable params: 608,261
Non-trainable params: 0

Figure 10: LSTM SIGMOID Model Summary

```
2/2 [==============================] - 0s 26ms/step
Multilabel Confusion Matrix:
[[[32  0]
  [ 0 13]]

 [[39  0]
  [ 0  6]]

 [[33  0]
  [ 0 12]]

 [[38  0]
  [ 1  6]]

 [[37  1]
  [ 0  7]]]

Accuracy Score: 0.9777777777777777

Classification Report:
              precision    recall  f1-score   support

       hello       1.00      1.00      1.00        13
  i love you       1.00      1.00      1.00         6
    will you       1.00      1.00      1.00        12
     tell me       1.00      0.86      0.92         7
   your name       0.88      1.00      0.93         7

    accuracy                           0.98        45
   macro avg       0.97      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45
```

Figure 11: LSTM SIGMOID Confusion Matrix/ Metrics Test Data

```
2/2 [==============================] - 0s 20ms/step
Multilabel Confusion Matrix for Validation Set:
[[[34  0]
  [ 0 11]]

 [[34  0]
  [ 0 11]]

 [[39  0]
  [ 0  6]]

 [[35  0]
  [ 1  9]]

 [[37  1]
  [ 0  7]]]

Accuracy Score for Validation Set: 0.9777777777777777

Classification Report for Validation Set:
              precision    recall  f1-score   support

       hello       1.00      1.00      1.00        11
  i love you       1.00      1.00      1.00        11
    will you       1.00      1.00      1.00         6
     tell me       1.00      0.90      0.95        10
   your name       0.88      1.00      0.93         7

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

Figure 12: LSTM SIGMOID Confusion Matrix/ Metrics Test Data

Bidirectional SIGMOID

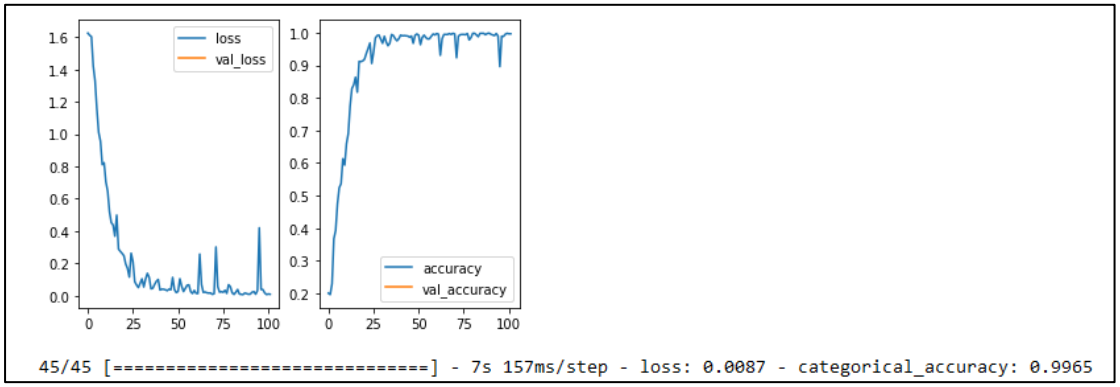


Figure 13: Bidirectional SIGMOID Plot Loss

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional_3 (Bidirectio  (None, 30, 128)          907264
 nal)

 dropout_50 (Dropout)        (None, 30, 128)           0

 bidirectional_4 (Bidirectio  (None, 30, 256)          263168
 nal)

 dropout_51 (Dropout)        (None, 30, 256)           0

 bidirectional_5 (Bidirectio  (None, 128)              164352
 nal)

 dropout_52 (Dropout)        (None, 128)               0

 dense_30 (Dense)            (None, 64)                8256

 dropout_53 (Dropout)        (None, 64)                0

 dense_31 (Dense)            (None, 32)                2080

 dropout_54 (Dropout)        (None, 32)                0

 dense_32 (Dense)            (None, 5)                 165

=================================================================
Total params: 1,345,285
Trainable params: 1,345,285
Non-trainable params: 0
```

Figure 14: Bidirectional SIGMOID Model Summary

```
2/2 [==============================] - 0s 26ms/step
Multilabel Confusion Matrix:
[[[33  0]
  [ 0 12]]

 [[36  0]
  [ 0  9]]

 [[38  0]
  [ 0  7]]

 [[38  0]
  [ 0  7]]

 [[35  0]
  [ 0 10]]]

Accuracy Score: 1.0

Classification Report:
              precision    recall  f1-score   support

       hello       1.00      1.00      1.00        12
  i love you       1.00      1.00      1.00         9
    will you       1.00      1.00      1.00         7
     tell me       1.00      1.00      1.00         7
   your name       1.00      1.00      1.00        10

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Figure 15: : Bidirectional SIGMOID Confusion Matrix/ Metrics Test Data

```
2/2 [==============================] - 0s 27ms/step
Multilabel Confusion Matrix for Validation Set:
[[[34  0]
  [ 0 11]]

 [[34  0]
  [ 0 11]]

 [[39  0]
  [ 0  6]]

 [[35  0]
  [ 0 10]]

 [[38  0]
  [ 0  7]]]

Accuracy Score for Validation Set: 1.0

Classification Report for Validation Set:
              precision    recall  f1-score   support

       hello       1.00      1.00      1.00        11
  i love you       1.00      1.00      1.00        11
    will you       1.00      1.00      1.00         6
     tell me       1.00      1.00      1.00        10
   your name       1.00      1.00      1.00         7

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Figure 16:: Bidirectional SIGMOID Confusion Matrix/ Metrics Validation Data

GRU SIGMOID

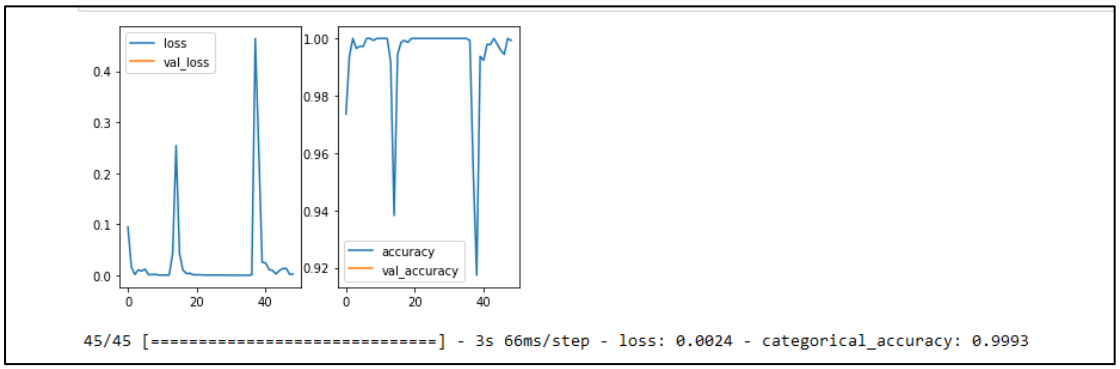


Figure 17: GRU SIGMOID Plot Loss

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_6 (GRU)                 (None, 30, 64)            340416

 dropout_55 (Dropout)        (None, 30, 64)            0

 gru_7 (GRU)                 (None, 30, 128)           74496

 dropout_56 (Dropout)        (None, 30, 128)           0

 gru_8 (GRU)                 (None, 64)                37248

 dropout_57 (Dropout)        (None, 64)                0

 dense_33 (Dense)            (None, 64)                4160

 dropout_58 (Dropout)        (None, 64)                0

 dense_34 (Dense)            (None, 32)                2080

 dropout_59 (Dropout)        (None, 32)                0

 dense_35 (Dense)            (None, 5)                 165

=================================================================
Total params: 458,565
Trainable params: 458,565
Non-trainable params: 0
_________________________________________________________________
```

Figure 18: GRU SIGMOID Model Summary

```
2/2 [==============================] - 0s 14ms/step
Multilabel Confusion Matrix:
[[[33  0]
  [ 0 12]]

 [[36  0]
  [ 0  9]]

 [[37  1]
  [ 0  7]]

 [[38  0]
  [ 1  6]]

 [[35  0]
  [ 0 10]]]

Accuracy Score: 0.9777777777777777

Classification Report:
              precision    recall  f1-score   support

       hello       1.00      1.00      1.00        12
  i love you       1.00      1.00      1.00         9
    will you       0.88      1.00      0.93         7
     tell me       1.00      0.86      0.92         7
   your name       1.00      1.00      1.00        10

    accuracy                           0.98        45
   macro avg       0.97      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45
```

Figure 19: : GRU SIGMOID Confusion Matrix/ Metrics Test Data

```
2/2 [==============================] - 0s 13ms/step
Multilabel Confusion Matrix for Validation Set:
[[[34  0]
  [ 0 11]]

 [[34  0]
  [ 0 11]]

 [[39  0]
  [ 0  6]]

 [[35  0]
  [ 0 10]]

 [[38  0]
  [ 0  7]]]

Accuracy Score for Validation Set: 1.0

Classification Report for Validation Set:
              precision    recall  f1-score   support

       hello       1.00      1.00      1.00        11
  i love you       1.00      1.00      1.00        11
    will you       1.00      1.00      1.00         6
     tell me       1.00      1.00      1.00        10
   your name       1.00      1.00      1.00         7

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Figure 20:: GRU SIGMOID Confusion Matrix/ Metrics Validation Data

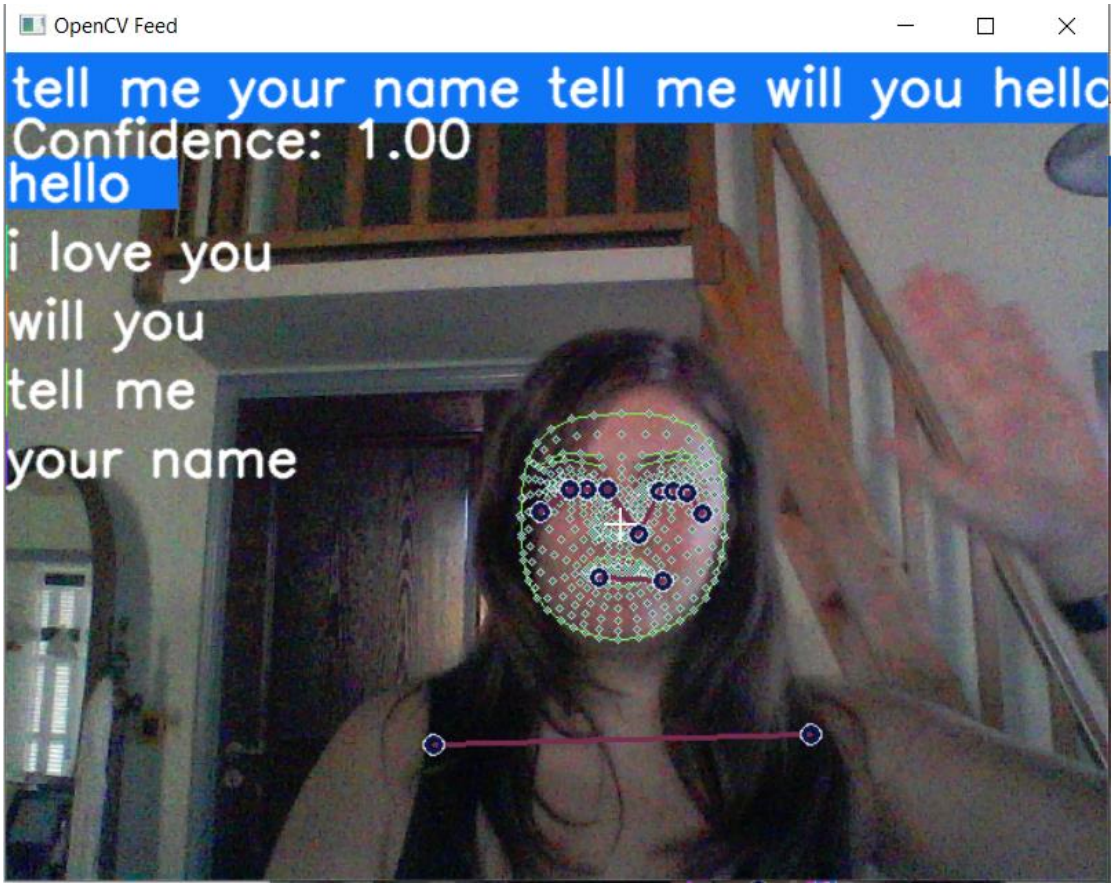


Figure 21: Class:Hello

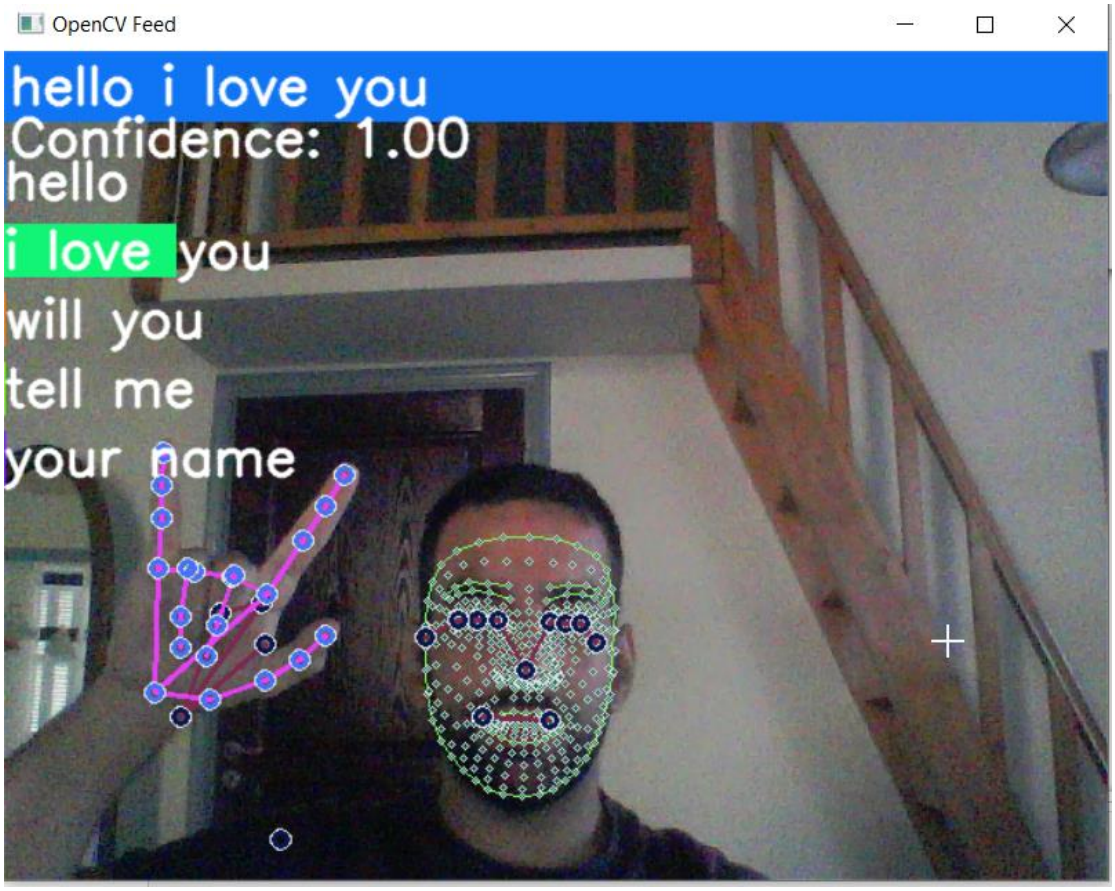


Figure 22: Class:"I love you"

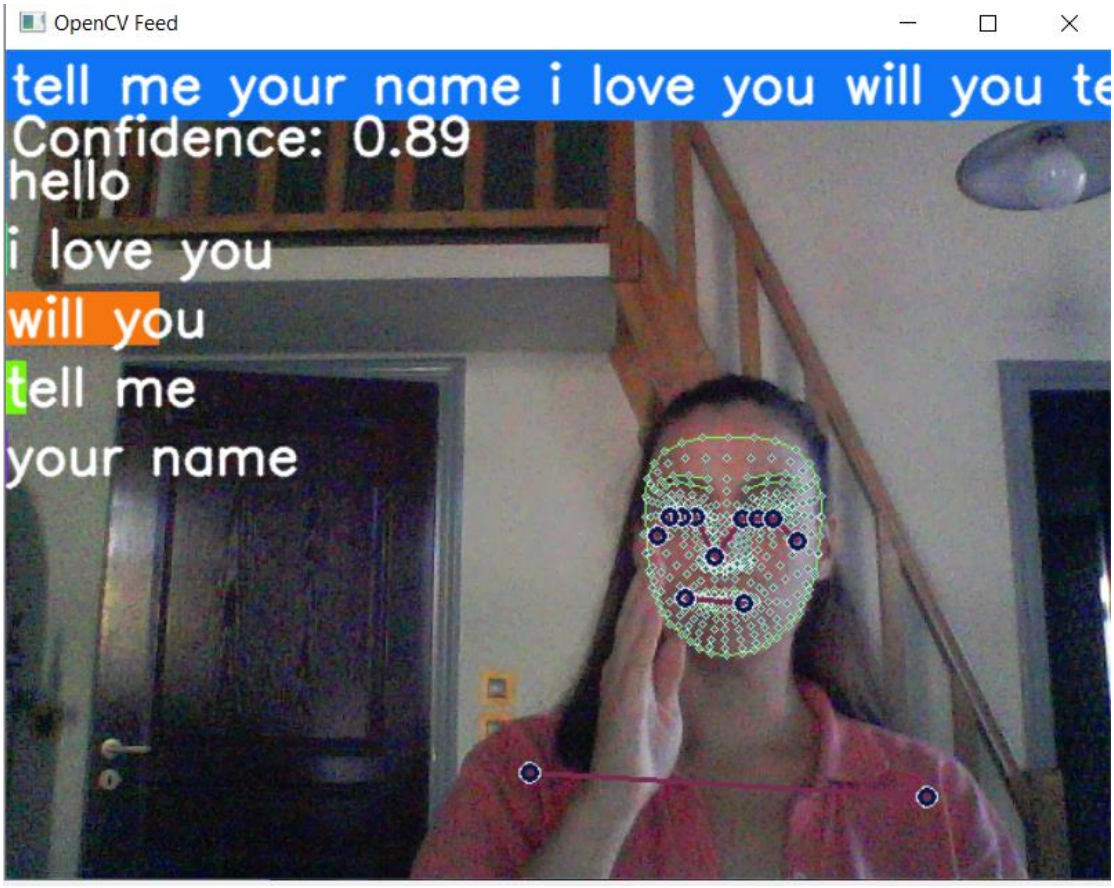


Figure 23: Class:"Will you"

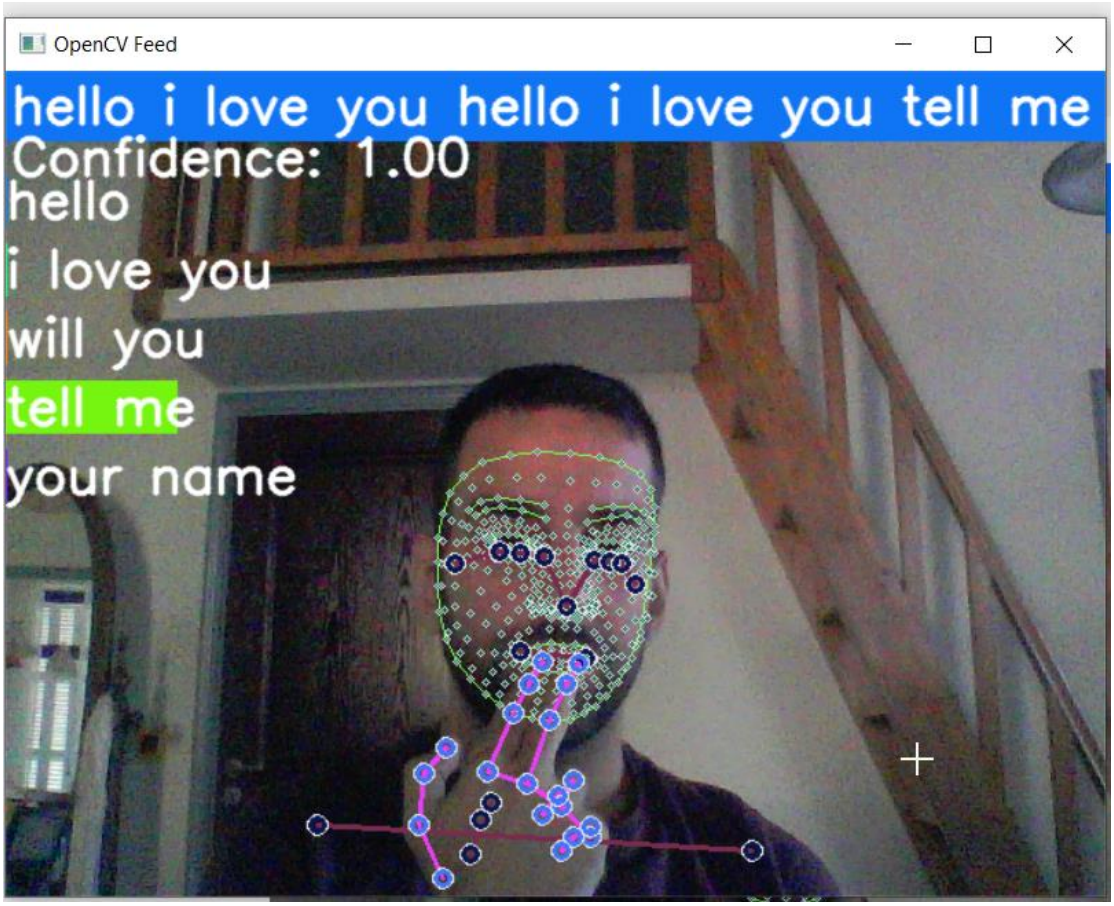


Figure 24: Class:"Tell me"

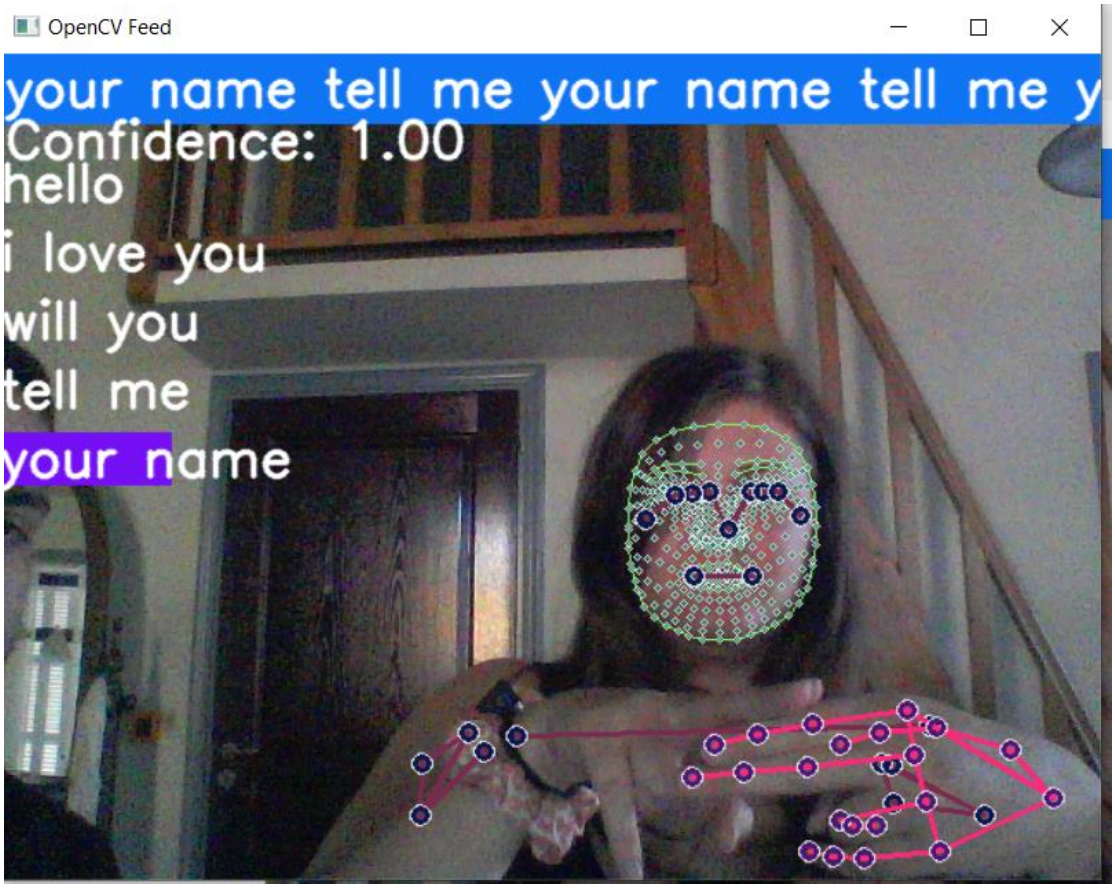


Figure 25: Class:"Your name"