A Deep Learning Approach of Translating Speech into 3D Hand Sign Language (ASL)

1st Kazi Mahathir Rahman
Department of Computer Science and Engineering
BRAC University
kazi.mahathir.rahman@g.bracu.ac.bd

2nd Naveed Imtiaz Nafis
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
naveed.imteaz.nafis@g.bracu.ac.bd

3rd Mohammad Al Rafi
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
mohammad.al.rafi@g.bracu.ac.bd

4th Mehedi Hasan Shahed
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
mehedi.hasan.shahed@g.bracu.ac.bd

5th Md. Farhan Sadik
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
md.farhan.sadik@g.bracu.ac.bd

***Abstract*—Automatic Sign Language Recognition (ASLR) is a rapidly advancing field that leverages computer vision and machine learning to interpret sign language gestures and convert them into text or spoken language. The goal is to facilitate communication between deaf and hearing individuals and to enhance accessibility in various domains. Each country typically has its sign language, such as American Sign Language (ASL) in the United States and British Sign Language (BSL) in the UK, with International Sign (IS) used in global contexts. While substantial research has focused on translating sign language to text, the reverse process of translating text to sign language still needs to be explored. This discrepancy highlights an essential gap in accessibility technologies for the deaf and hard-of-hearing communities. Text-to-sign language translation involves converting written or spoken language into corresponding sign language gestures, which is inherently complex due to sign language's rich, visual, and spatial nature. Our research aims to develop a novel system for converting spoken language into 3D hand sign language, bridging the communication gap between the hearing and deaf communities. This innovative approach involves two main components: automatic speech recognition (ASR) and 3D hand sign generation.**

***Index Terms*—3D generation; American sign language (ASL); Text-to-sign langugae; Automatic Speech recognition (ASR)**

I. Introduction

A. Thought behind using 3D ASL with NLP and Deep learning

Imagine standing in a room full of people, and you hear sounds but do not know what those sounds are. This is the story of millions of deaf and hard-of-hearing individuals. However, being proficient in sign language, American Sign Language(ASL) is rich. Now, imagine a technology that can fill this gap, using speech-to-hand translation to convert spoken words into live, 3D hand signs in Sign Language. We aim to unfold the promising cross-section between deep learning and natural language processing (NLP) and how this makes the above vision possible to convert speech into 3D ASL and enhance access to the Deaf and Hard of Hearing(DHH).

The fields of deep learning and natural language processing have advanced significantly to change the dynamics of human-computer interaction and create new doors to building communication systems for the hearing-impaired. ASL is a complicated visual-spatial language that uses the hand(s), facial expressions, and posture of the body for establishing contextual meanings in sign-stylized and lingualised forms. According to Bragg et al. [1] (2019), there exist a significant amount of limitations that are connected with real-time translation, especially in the context of spoken language and the accessibility of media for people with DHH. These limitations can be reduced by applying deep learning methodologies to convert speech to 3D ASL to improve communication.

Additionally, using 3D modeling to translate ASL is more beneficial than a mere 2D representation because this technique is more accurate with regard to sign language's complex and dynamic structure. 3D models also allow us to create animations, video games, augmented reality, and 3D printing. In their journal, Ebling et al. [2] (2015) highlight that this 3D(3D avatar) modeling approach can ensure accurate hand gestures and facial expressions for accurate and meaningful translation. Besides, by using deep learning and advanced NLP technology, a 3D translation system will enrich the quality of life for the people of the DHH community.

However, with technological advancements in recent times, there has been a growing need for a more elaborate general system linking the DHH community with general public communication needs. New developments in the 2D sign language recognition format are progressing, yet 3D modeling appears to provide a more precise alternative, particularly for a highly spatial language like ASL. According to Ebling et al. [2] (2015), a 3-D avatar can obtain signals from subtle facial features and unique hand gestures essential for the accuracy of sign language, which a 2D model fails to depict. Chakladar et al. [3] (2021) affirm that the combination of 3D models with NLP might change the whole outlook of symbolization of ASL through dynamic hand gestures during the transition from texts or voice.

Likewise, cost-effective and rapid interaction in translating spoken language into ASL poses a significant challenge to 3D deep learning models when coupled with NLP. Models such as BERT(Bidirectional Encoder Representations from Transformers) and LSTM(Long Short-Term Memory), already giving validation to NLP, do help in developing an identification and deciphering of the gestures of the sign language for the language spoken. Making use of NLP helps capture the grammar and context such that prediction accuracy for ASL signs increases, while the 3D models provide a spatial and visual medium to represent those signs.

Nevertheless, the modernistic advanced innovations of breakthrough speech recognition systems, like Whisper AI, increase the applicability of this system in various languages and dialects and make it a more robust and inclusive solution. Studies such as those by Saunders et al. [4] (2020) point to their usage of transformer-based systems to deliver both sign accuracy and gesture fluidity in real-time.

B. Research Problems

First of all, ASL translation technologies are facing extreme challenges, way beyond mere visual input and recognition systems, when it comes to ensuring real-time communication between deaf and hearing communities. Apart from having human interpreters, they involve computer recognition that goes beyond the rudiments of the anthropometric system. Sign language encompasses many levels of complexity that represent a continuous disjoint with the computer world, existing in architectures of their own. Because ASL uses hand gestures, facial interpretation, and body movement simultaneously, such complexities remain highly limiting to the usability of most of the existing technology for real-time application and communication interfaces-and they play a direct role in making it difficult for 2-D models of ASL to capture this depth. As pointed out by Konstantinidis et al. [5] (2018), deep learning approaches are still operating in the shallow zone of meeting the most basic of human gestures.

Secondly, years of effort have blended Natural Language Processing (NLP) with deep learning to open new doors toward serving and improving sign language translation. However, most studies have focused purely on recognition from visual input, and an alarming shortage of studies have focused on the translation of spoken or written language into 3D ASL gestures. With the massive advances made in speech-to-text technologies, sign language translation is still in a developing phase and estimably fares poorly about the complexities involved in interpreting ASL. The absence of skeletal data of 3D ASL gestures presents yet another hurdle to building sufficiently trained models to translate spoken language into ASL in three-dimensional terrain (Ebling Glauert, 2015) [2].

Moreover, the real-time process of translation using ASL is further impeded by the computation and high-level learning algorithms extensively used for processing. The need for real-time translation also comes to areas such as education, public services, and broadcasting, where one could easily work themselves into a backup with translation delays and a slow workflow. According to Bragg et al. [1] (2019), a sophisticated level of accessibility technologies is still far from being in place when it comes to providing perfect and flawless, real-time ASL translation. For this, it is vitally important to develop a system that can process thereby translating complex American Sign Language-oriented sign expressions at a fairly fast rate while maintaining authenticity and fluidity.

Another important issue is that current systems do not know how to give context to ASL gestures. ASL is not simply English-translated word for word. Therefore, ASL has a unique nature regarding its grammatical and syntactical properties. Most existing models treat ASL more or less as a simple mode of spoken language translation. This segmenting character can cause meaning displacement, especially in more refined or context-centric phrasing. The resolution of this problem seems apparent in that NLP models such as BERT and LSTM could be integrated to provide translations reflective in the context during cases of a real-time translation of ASL. These models, after being trained with relevant data, would encapsulate a certain amount of situational distinction in providing translations rather than the old methods (Stoll et al., 2019) [6].

As a result, this research aims to answer the question of how effective is their combination of state-of-the-art NLP models with 3D deep learning techniques for improvements in real-time ASL translation.

Finally, this research will also focus on the appraisal of the new system's capacity regarding real-world implementations. The core metrics awaiting examination include accuracy, latency, and user satisfaction. WLASL and RWTH-PHOENIX-Weather will be datasets against which the effectiveness is quantified. Ultimately, the more generic purpose lies in contributing to building accessible communication technologies that create equality and inclusivity for the DHH community.

C. Aims and Objectives

This research develops a highly robust 3D ASL recognition system driven by NLP and deep learning to convert spoken or written language into reasonably accurate 3D hand signs. The main objectives of this research are as follows:

- To explore the applicability of Natural Language Processing (NLP) and deep learning techniques for American Sign Language (ASL) gesture prediction and real-time translation, focusing on models like BERT and LSTM.
- To offer a dataset of 3D ASL key points, leveraging existing tools such as MediaPipe, OpenPose, and FrankMocap for data extraction.
- To create a predictive model that translates spoken language into real 3D hand signs, with seamless integration into platforms like Unity for real-time interaction.
- To validate this model using real-world datasets, such as WLASL (Word-Level ASL Dataset) and RWTH-PHOENIX-Weather, to ensure accuracy and applicability.
- To highlight the system's usability and accessibility in the real world, addressing the needs of the Deaf and Hard of Hearing (DHH) community.

II. Literature Review

Recent years have seen notable development in sign language recognition systems, mostly due to breakthroughs in computer vision technology. In 1998, Ouhyoung [7] invented a system that has the capability to constantly and instantly identify sign language in real-time. This system acknowledges signs directly from video streams without segmentation. It breaks down signs into essential elements like postures and motions, using hidden Markov models (HMMs) to identify them and statistically analyze the gesture for continuous recognition. Infantino et al. [8] (2007) present a system for sign language recognition (SLR) that uses common sense knowledge. Their system goes beyond individual signs, incorporating context to improve accuracy. It employs a multi-level approach: recognizing signs, building sentence structure, and using commonsense reasoning to check for inconsistencies. This improves the Continous Sign Language Recognition (CSLR). Verma et al. [9] (2007) explore combining information from different sources (hand shape, facial expressions) to improve continuous sign language recognition (CSLR). They investigate three fusion techniques: early (combining features before classification), late (combining decisions from separate classifiers), and synchronous hidden Markov model (HMM) integration. Gao et al. [10], [11] (2004) introduce transition movement models (TMMs) to tackle large vocabulary continuous sign language recognition (CSLR). Existing methods need help with the many signs and seamless transitions between them. TMMs focus on modeling these transitions, improving recognition accuracy by capturing the nuances of movement between signs. Everything changed after the advancement of the deep learning approach. In 2016, Koller et al. [12] (2016) addressed training a CNN for hand images with limited data. To tackle this issue, they integrate a Convolutional Neural Network (CNN) into an iterative Expectation-Maximization (EM) process. The CNN iteratively improves its classification by using weak video-level labels, hence enhancing frame-by-frame annotations with each iteration. Pu et al. [13] (2019) propose a new method for continuous sign language recognition (CSLR) by using an iterative alignment network. Their approach addresses CSLR by iteratively improving two components: a 3D-ResNet for learning features and an encoder-decoder with CTC for predicting sequences. A soft-DTW constraint guides training by aligning predicted signs with the ground truth.Konstantinidis et al. [14] (2018) suggest a deep learning technique for recognizing sign language. This method involves analyzing video data, which includes both images and motion, as well as skeletal data, which captures body stance performed using Convolutional Neural Networks (CNN). The system uses separate CNN networks for video and skeletal features, then combines them using various fusion techniques. Yuan et al. [15] (2022) propose DeepSign, an advanced deep learning system that utilizes Convolutional Neural Networks (CNNs) to process sign language. DeepSign focuses on two tasks: detecting the presence of sign language in videos and identifying the specific signs being used. This unified CNN-based approach streamlines processing compared to separate models for each task. However, as our method is the opposite of sign language recognition, we want to generate 3D hand signs from voice recognition. First of all, we have to convert the voice into text.

In the late 20th century, many researchers started working with signal processing, specifically converting voice signals to handwritten text. HARPY, [16] a significant 1970s speech recognition system, built upon prior systems (Hearsay-1, Dragon) to understand key design choices. Unlike them, HARPY used finite state transition networks for knowledge representation. It employed a beam search strategy to explore promising sound combinations and segmentation for efficiency. The system also incorporated semi-automatic learning of pronunciations and sound representations from training data. Kumar et al. [17] (2009) propose an objective method to assess stuttering severity. Traditionally, assessments are subjective, leading to inconsistency. This study tackles this issue by using Mel-Frequency Cepstral Coefficients (MFCCs), a widely used method for extracting features in speech processing, to understand the characteristic's of speech. They likely segment speech into syllables, extract MFCCs, and potentially use machine learning to classify fluent versus stuttered segments. Ravikumar et al. [18] (2008) address an objective assessment of stuttering severity in read speech.

Traditionally, assessments are subjective. This work targets a common stuttering characteristic - syllable repetitions. They propose the implementation of an automated detection method that leverages Mel-Frequency Cepstral Coefficients (MFCCs) to extract features and employs Dynamic Time Warping (DTW) to compare syllables and identify repeats. This objective approach using MFCCs and DTW offers consistent evaluation compared to subjective methods, potentially improving stuttering assessment. Mitrovic et al. [19] (2009) highlight the significance of feature selection in environmental sound recognition (ESR) because of the diverse range of sounds and the possible constraints of employing generic audio features. The paper highlights the value of statistical analysis (PCA) in identifying redundant features and guiding selection. Chen et al. [20] (2021) introduce WavLM, a pre-training approach for various speech-processing tasks. WavLM utilizes self-supervised learning, where a large model is trained on masked speech prediction (predicting missing parts of speech) and denoising (removing noise). Watanabe et al. (2021) [21] introduces UniSpeech, a method for learning speech representations that addresses the challenge of limited labeled data. Traditionally, training speech recognition models require a lot of labeled data, which can be expensive. UniSpeech tackles this by proposing a unified approach that leverages labeled and unlabeled data. It uses a multi-task learning framework, combining supervised learning on labeled data (classifying speech sounds) with self-supervised learning on unlabeled data (using contrastive learning). After machine learning, different deep-learning approaches (CNN, LSTM, RNN, etc.) were used for automatic signal processing. Liu et al. [22] (2018) address speech recognition in noisy environments. Their approach involves using a Long Short-Term Memory (LSTM) model to improve speech signals before inputting them into a speech recognition system. LSTM networks are adept at handling sequential data, allowing them to identify and remove noise patterns within the speech. Zhang et al. [23] (2019) propose an innovative approach for Automatic Speech Recognition (ASR) by integrating Convolutional Neural Networks (CNNs) with Bidirectional Long Short-Term Memory (LSTM) networks. CNNs are skilled at extracting features from speech signals, while LSTMs excel at capturing context within sequences. This hybrid approach merges these strengths. CNNs likely extract low-level features, and LSTMs then analyze them in both directions to grasp context and long-term dependencies. Li et al. [24] (2020) propose ContextNet, a deep learning architecture for ASR that combines CNNs, RNNs, and transducers. While CNNs are powerful for ASR, they can struggle with capturing global context in speech. ContextNet addresses this by incorporating a mechanism using squeeze-and-excitation modules to inject global context information into the CNN layers. The latest approaches are transformer-based approaches. Turchi et al. [25] (2021) propose Speechformer, a novel approach to direct speech translation that addresses information loss. Traditionally, speech is compressed before translation, leading to missed information. Speechformer utilizes a memory-efficient Transformer model to process raw audio signals at a higher resolution. This preserves more linguistic details, potentially improving translation accuracy and fluency. The study conducted by Baevski et al. [26] was published in 2022. Presented is data2vec, a versatile system for self-supervised learning that may be used across several domains such as audio, vision, and language. Traditionally, self-supervised learning methods are modality-specific, requiring different approaches for each data type. data2vec tackles this by proposing a unified method using masked language modeling with Transformers. Based on context, the model predicts masked parts of the input data (words in text, regions in images). Wang et al. [27] (investigate the year) propose VALL-E, a novel approach for zero-shot text-to-speech synthesis (TTS). Unlike traditional methods that directly model speech waveforms, VALL-E utilized TTS as a language modeling task. It predicts a sequence of codes representing speech features conditioned on the input text and a short audio clip from an unseen speaker.

Text-to-3D generation is a rapidly evolving field in artificial intelligence (AI) that aims to create 3D models from textual descriptions. GET3D [28] (2022) presents a novel approach to creating high-quality 3D textured shapes using 2D images. It surpasses prior methods by creating 3D models with intricate details, complex structures, and realistic textures. Dream3D [29] (2022) addresses issues of inaccurate 3D generation from text descriptions in prior methods. The process involves the creation of a 3D form in two steps: first, a text-to-image diffusion model creates a 2D picture that represents the 3D object. This image is then used to develop a preliminary 3D shape. DREAMFUSION [30] (2022) investigates the process of creating three-dimensional models from text by using a pre-trained two-dimensional text-to-image diffusion model. The system employs Score Distillation Sampling (SDS) to optimize a 3D scene representation (NeRF) according to the required 2D picture outputs specified in the text description. Magic3D [31] (2022) presents a novel method for creating high-resolution 3D content from text. It surpasses prior methods like DreamFusion by generating detailed 3D mesh models with textures. Magic3D is faster than previous techniques while creating higher-quality models. DreamBooth3D [32] (2023) addresses personalizing text-to-3D generation by leveraging a few casual images (3-6) of the desired subject. It combines DreamBooth (image personalization) with DreamFusion [30] to overcome the limitations of prior methods. Through a 3-stage optimization process, DreamBooth3D generates 3D models that are consistent with the subject's appearance and adhere to the text description, allowing for the creation of personalized 3D models with various details based on text prompts. Fantasia3D [33] (2023) addresses text-to-3D generation by proposing a novel method that disentangles geometry and appearance during learning. This means it separates learning the 3D shape (geometry) from learning surface properties (appearance). DiT-3D [34] model, a method for generating 3D shapes. Unlike prior U-Net-based methods, it leverages diffusion transformers and operates directly on voxelized point clouds. DiT-3D utilizes transformers without any modifications, including 3D positional and patch embeddings as well as 3D window attention to improve efficiency.

Specific systems are designed to automatically convert spoken or written text into precise representations of a particular sign language, such as American Sign Language (ASL). This emphasizes the ongoing development as a result of generative AI advancements. Camgoz et al. [35] (2018) propose an innovative method for Neural Sign Language Translation (NSLT) by using Neural Machine Translation (NMT) approaches. This method directly converts sign language images and videos into spoken language phrases, removing the need for distinct sign recognition. Liu et al. [36] (2018) propose an LSTM neural network to enhance speech recognition in noisy environments. This method preprocesses speech signals to remove noise patterns before feeding them into a speech recognition system. It leverages the ability of LSTMs to capture sequential data and distinguish between noise and speech. Stoll et al. [37] (2019) present Text2Sign, a system that converts spoken English into sign language small video clips. Addressing the limitations of data-intensive methods, Text2Sign utilizes a two-stage approach. First, it translates spoken sentences into sign sequences using Neural Machine Translation (NMT). Then, it leverages a Motion Graph and a generative model to translate these signs into realistic videos. Saunders et al. [38] (2020) address automatic sign language production limitations. Previous methods focused mainly on hand movements, neglecting facial expressions and mouth motions. This paper proposes Adversarial Training with a transformer-based generator and a discriminator conditioned on the spoken language. The discriminator acts as an ”adversary” to assess the realism of generated signs. Chaudhary et al. [39] (2023) address a limitation in sign language translation where research often focused only on recognition (sign language to text). The authors present SignNet II, a transformer-based model designed for bidirectional translation. SignNet II can recognize sign language from visuals and generate sign language from text. It achieves this through a two-branched approach: one for sign recognition and another for text-to-sign generation. Wang et al. [40] (2022) propose a new method for generating 3D sign language directly from text, addressing the limitations of prior methods that require complex data or architectures. Their approach, called ”There and Back Again,” utilizes back-translation. Initially, a model is employed to convert text into 2D poses. These poses then train a separate model to convert them back into text. Ultimately, the model undergoes fine-tuning to produce 3D sign language sequences based on the input text.

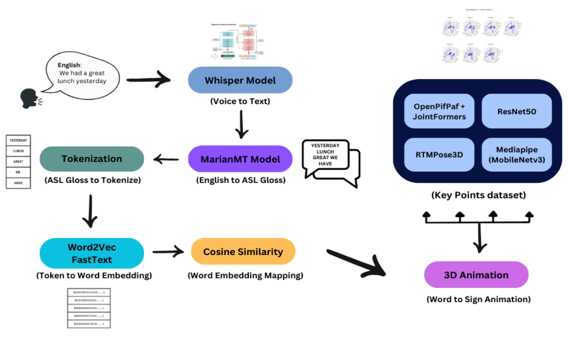


Fig. 1. Architecture Proposed System

III. Methodology

IV. Architecture of the Proposed System

Our system translates spoken English into 3D animated American Sign Language (ASL). First, speech is converted to text using Whisper. This text is then broken down and converted into an ASL gloss (written representation of signs) using advanced language models and word embeddings. Next, these gloss representations are mapped to 3D key points representing hand and body movements using motion capture data and pose estimation model datasets. Finally, these 3D keypoints drive a rigged 3D avatar, generating a real-time animation of the corresponding ASL signs.

V. Used Architecture for Speech to ASL Conversion Pipeline

In our research, we developed a comprehensive pipeline for converting spoken language into American Sign Language (ASL). The details of them are given below:

A. *Voice-to-Text Conversion*

The process starts with converting speech into textual form, ultimately leading to 3D hand sign language output. For this, we will be using an OpenAI-developed software tool named “Whisper”. Whisper is a state-of-the-art, transformer-based automatic speech recognition(ASR) system trained on 680000 hours of multilingual and multitask supervised data that performs effectively under varying speech conditions within a noisy environment, with high variation in accents as well as, multiple languages. This makes Whisper the most suitable model for a real-world speech-to-text conversion.

The Whisper architecture is a simple, end-to-end system based on an encoder-decoder Transformer. The system accepts 30-second audio segments that are converted to “log-Mel” spectrograms before passing them into the encoder processing. The decoder generates text predictions along with specific tokens which lead operations including language detection phrase timestamp tagging and multilingual transcription and English translation outputs. Whisper demonstrates better zero-shot performance across multiple datasets leading to 50% fewer errors compared to standard models like ”LibriSpeech”. Whisper’s training data includes one-third of non-English content which demonstrates effectiveness in multilingual tasks primarily through spoken language to English translation [41].

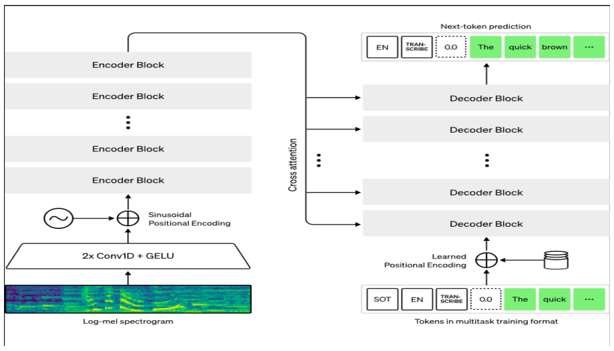


Fig. 2. Whisper Architecture [41]

The audio system applies normalization and segmentation into smaller frames processing to audio data making it ready for efficient work. Then Whisper's tokenizer converts these frames into tokens to make the texts temporal align with the original speech. This alignment is critical for subsequence stages because the subsequent stages derive their operations from text data.

B. Text to ASL Gloss Conversion

Translating English text into ASL gloss performs multiple mappings of language structure onto sign language elements. It consists of two primary processes: tokenization and embedding.

1) ASL Tokenization: The implemented custom tokenization process extends past basic whitespace separation when creating tokens. The steps include:

1) **Token Segmentation:** This splits sentences into smaller units based on punctuation markers and whitespace. Token segmentation divides elements into separate units through which each word becomes an independent token.
2) **Vocabulary Creation:** When mapping tokens to IDs the model uses the tokens' frequency of occurrence in the training data. The utility of particular expressions determines their assigned numeric identifier based on their common usage. For example, let's assume the word "me" is used 3000 times in a paragraph, and "you" is used 1000 times. Thus "me" will receive a lower ID than "you"(Here, the lower the ID higher the prominence it will get). In this process, we obtain a total of 68,000 unique vocabulary tokens.
3) **Padding and Truncation:** They are preprocessing techniques used to standardize input sequence lengths in machine learning models. The padding adds special tokens, such as [null] to shorter sentences to match the required length. For instance, if the fixed length is 6, the sentence "Hand Sign" becomes [Hand, Sign, [null], [null], [null], [null]]. Truncation shortens longer sequences by cutting off excess words beyond the fixed length. For example, the sentence "Understanding hand sign requires effort and patience to master" is truncated to [Understanding, hand, sign, requires, effort, and] if the length is set to 6. These adjustments ensure consistent input dimensions across all data samples. In our case, we use a padding length of 30 to standardize input sequences.

Using this tokenization process, we train translation models, including T5 BART, and MarianMT, which regulate ASL gloss datasets. Using this tokenization process, we train translation models including T5 and BART, controlled by ASL gloss datasets. Both models rely on pre-trained transformer architectures to generate sequences conditional on the input text. T5 takes a unified framework for processing multiple text-based tasks, while BART reconstructs input sequences with added noise to improve its robustness by translating between English and ASL gloss [42].

2) Word Embedding: To further refine the conversion process, we utilize four embedding models:

a) Word2Vec (CBOW): The Continuous Bag of Words is a method in Word2Vec introduced by Mikolov et al. in 2013 [43]. The CBOW is one of the methods of word embedding and it predicts and learns target words from the surrounding context. The system utilizes defined words through co-occurrence patterns analysis to build compact vector models that reveal global word semantic relations across contexts. The characteristic of the CBOW model to identify contextual relationships finds special value when translating English words to ASL gloss since semantic precision requires deep context understanding. For example, through vocabulary embeddings, it will detect that "happy" and "joyful" are similar to each other in semantic space which allows for more precise gloss generation.

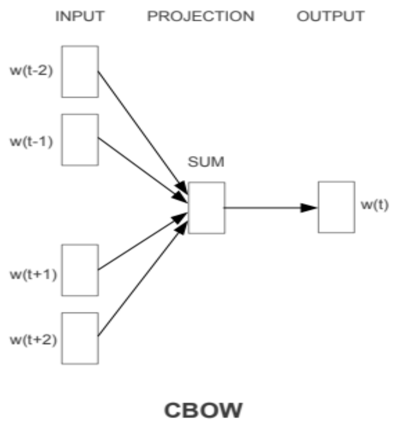


Fig. 3. CBOW model architecture [43]

.

The CBOW model predicts a target word w_t based on the surrounding context words $(w_{t-n}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+n})$. By analyzing the co-occurrence patterns of words within a fixed window size (n), CBOW creates dense vector representations that capture global semantic relationships. The function for CBOW is below:

$$L_{\text{CBOW}} = \prod_{t=1}^{T} P(w_t \mid w_{t-n}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+n}),$$

where $P(w_t \mid \cdot)$ is modeled using a softmax function:

$$P(w_t \mid \cdot) = \frac{\exp\left(\mathbf{v}_{w_t}^{\top} \mathbf{v}_c\right)}{\sum_{w \in V} \exp\left(\mathbf{v}_w^{\top} \mathbf{v}_c\right)}.$$

Here, $\mathbf{v}_{w_t}$ represents the target word vector, $\mathbf{v}_c$ is the context vector, and V is the vocabulary size.

The implementation of aggregated context information makes CBOW efficient for large-scale datasets hence reducing computational time during training. The system achieves the highest effectiveness during training of extensive ASL datasets containing diverse vocabulary entries. CBOW embeddings demonstrate generalization capabilities that enable tuning-specific gloss datasets toward better ASL syntactic and semantic alignment. CBOW's robust embedding method and gloss mapping ambiguities prove essential for speech-to-ASL pipelines.

b) Word2Vec (Skip-gram): The Skip-gram model shows different behavior from CBOW by predicting contextual words that surround a specific target input word while searching for rare word connections. The method stands out in processing rare vocabulary items because they appear frequently in ASL gloss contexts. Skip-gram demonstrates an exceptional ability to recognize professional terminology and native phrases which leads to improved ASL domain representation [44]. To support fine-grained ASL interpretation Skip-grams produce neural representations that maintain both syntactical and semantic relations among rare words.

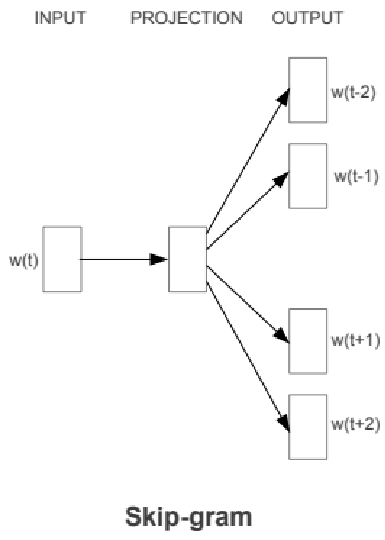


Fig. 4. Skip-gram model architecture [44]

The Skip-gram model predicts the surrounding context words $(w_{t-n}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+n})$ based on a given target word w_t. The objective function is:

$$L_{\text{Skip-gram}} = \prod_{t=1}^{T} \prod_{-n \leq j \leq n, j \neq 0} P(w_{t+j} \mid w_t),$$

where $P(w_{t+j} \mid w_t)$ is similarly modeled by the softmax function:

$$P(w_{t+j} \mid w_t) = \frac{\exp\left(\mathbf{v}_{w_{t+j}}^{\top} \mathbf{v}_{w_t}\right)}{\sum_{w \in V} \exp\left(\mathbf{v}_w^{\top} \mathbf{v}_{w_t}\right)}.$$

Skip-gram processes information at a higher computational cost than CBOW yet its capability to work with infrequent words proves valuable when gloss datasets consist primarily of rare terminology. When trained with multilingual 'corpora' Skip-gram embeddings create a semantically robust system for generating ASL glosses across different languages. By combining with BERT and FastText models Skip-gram improves rare word associations which helps boost the precision of English-to-ASL gloss mapping systems.

c) FastText: FastText introduced by (Joulin et al., 2016) [45] builds word representations from multiple character sequences which enhances its ability to embed subword information. The distinct methodology of FastText breaks down words into character-level fragments that deliver successful processing of out-of-vocabulary expressions. The FastText model brings stable mappings for vocabularies that have infrequent use or were newly created because standard word embeddings do not include them. FastText processes terms including "cybersecurity" and "metaverse" by automatically generating relevant word embeddings that lack presence in the training database.

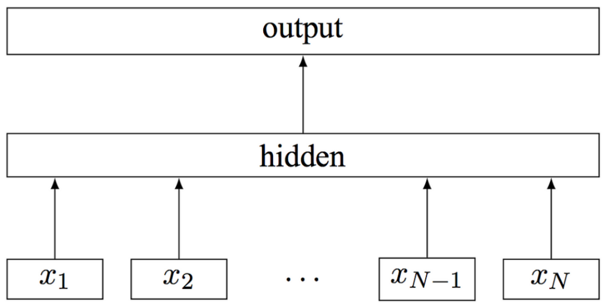


Fig. 5. FastText model architecture [45]

.

FastText represents words as a sum of their subword n-gram embeddings. A word w is decomposed into a set of character n-grams ($g \in G_w$), and its vector is computed as:

$$\mathbf{v}_w = \sum_{g \in G_w} \mathbf{v}_g,$$

where:

- $\mathbf{v}_w$: Word vector for word w.
- G_w: Set of subword n-grams for w.
- $\mathbf{v}_g$: Vector representation for subword g.
- $|G_w|$: Number of subword n-grams in w.

The morphological variation detection capabilities of FastText prove valuable when generating ASL glosses because word forms adjust according to contextual use. By including subword components in the model FastText constructs a space where morphologically related words like ”run,” ”running” and ”runner” stay in close proximity. The ability of FastText to capture morphological variations enables it to generate consistent glosses particularly crucial for real-time applications that handle frequent new or complex vocabulary.

d) BERT (Bidirectional Encoder Representations from Transformers): BERT was first introduced in 2018 by Devlin et al [46]. which transformed word embeddings through its addition of Bidirectional Encoder Representations from Transformers training methodology. The text-processing approach of BERT functions differently from sequential models and performs analysis of word context from both left and right directions because it enables deep identification of semantic and syntactic information. Through bidirectional training, BERT embeds sentences fully to maintain accurate English phrase to ASL gloss matchings. BERT demonstrates the ability to recognize two different senses of ”bank” between ”river bank” and ”financial bank” which proves fundamental to creating high-quality glosses.

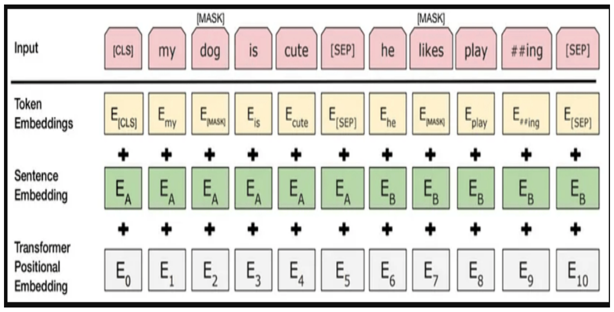


Fig. 6. BERT input representation [46]

.

BERT uses a bidirectional transformer architecture that predicts missing words in a sentence. Its masked language modeling (MLM) function is:

$$L_{\text{MLM}} = \sum_{t \in M} \log P(w_t \mid \text{context}),$$

where M is the set of masked positions, and $P(w_t \mid \text{context})$ is computed using the transformer's self-attention mechanism:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V.$$

Here:

- Q: Query matrix
- K: Key matrix
- V: Value matrix
- d_k: Dimension of the keys

BERT starts its operation with a massive text corpus training but developers can fine-tune its parameter values to match the linguistic features found in gloss annotations when creating ASL gloss generation functionality. The transformer-based architecture applies self-attention processes to measure word significance against other sentences through its network. The embedding computation method ensures both lexical meaning and grammatical structure remain within the output information.

The embedded language originated from these models transform English words into their corresponding ASL glosses. We employ cosine similarity methods to identify 2,000 unique words that form the basis for gesture generation.

3) Transformer-Based Models for ASL Gloss Conversion:

a) BART (Bidirectional and Auto-Regressive Transformers): BART was first introduced by Lewis et al. (2020) [42] which operates as a sequence-to-sequence transformer model for text reconstruction work. BART applies pretraining by processing corrupted input text. It inserts masked tokens into the input. It also deletes some input words. Additionally, it rearranges sentences in the input. The model then learns to reconstruct the original inputs. Through its training strategy, BART brings together knowledge of both text semantics and structure which results in excellent resilience against ambiguous or inconsistent inputs. The system benefits from robust operation when generating ASL glosses because it handles transcription errors from Whisper speech-to-text systems.

BART maintains a bidirectional encoder paired with an autoregressive decoder as its basic framework. The encoder establishes full attention to each sequence token during processing which results in contextual embedding calculations that capture global dependency relationships. Among these embeddings, the decoder predicts the output sequence through token-by-token autoregressive attention operations. BART achieves successful English-to-ASL translation. It uses bidirectional encoding combined with autoregressive decoding. This approach helps it detect both sentences and linguistic specifics effectively.

BART obtains improved performance for ASL gloss generation through a joint training process on clean and noisy datasets to support real-world uses. By using this approach the model maintains precise translation functionality for instances that present notable grammatical divergence between the two languages. During reconstruction, BART aligns generated ASL glosses with the original semantic content of input text.

b) T5 (Text-to-Text Transfer Transformer): T5 was introduced by Researchers Raffel et al. (2019) [48] and developed by Google AI as a transformer framework which performs natural language processing (NLP) tasks through a text-to-text paradigm. The T5 architecture treats every task as a text-generation problem. It maintains a consistent learning pathway for all tasks. The flexibility of T5 lets the model perform ASL gloss generation tasks while retaining total focus on important linguistic elements within texts including syntax and context.

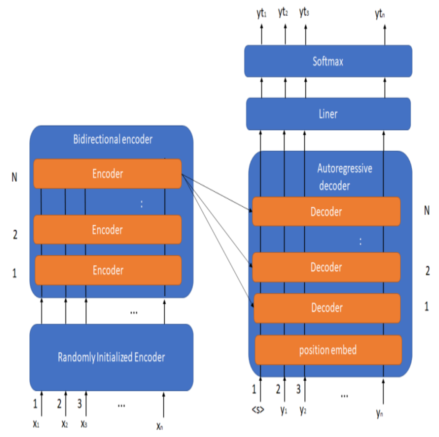


Fig. 7. BART Model Architecture [47]

.

Within its structure, the T5 model executes summarization tasks alongside translation operations successfully without any distinction.

During pretraining, T5 utilizes C4(Colossal Clean Crawled Corpus) which handles massive web-crawled text information to enhance model capability across diverse domains. Within its encoder-decoder structure, T5 applies self-attention processing to manage sequence operations effectively. The encoder processes input sequences and transforms them into dense contextual embeddings. These embeddings allow the decoder to produce output sequences by referring to the encoder's states. The encoder uses bidirectional attention, which helps the model understand the full context. This enables effective translation of English text into ASL gloss sequences. During pretraining, T5 uses span corruption by blocking parts of the text. This helps the model learn to reconstruct the hidden sections and improve its contextual understanding.

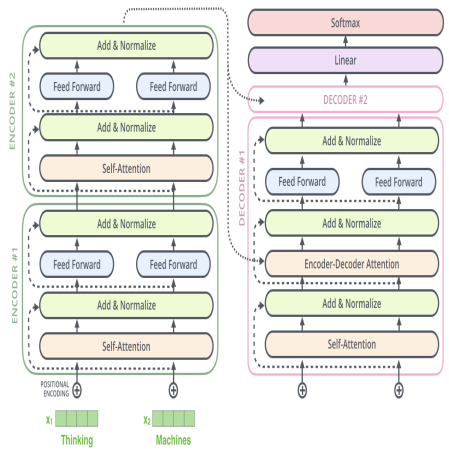


Fig. 8. T5: Text-to-Text Transfer Transformers Structure [49]

.

For generating ASL glosses from text with T5 an adjustment process must refine pre-trained parameters by using datasets tailored to ASL language requirements. T5 learns specific grammatical structures and syntactical patterns of ASL at this stage which drives its ability to produce correct glosses. Model efficiency increases through padding and truncation methods which normalize input sequence lengths.

c) MarianMT: The transformer-based system MarianMT operates as a powerful machine translation solution with impressive performance efficiency. MarianMT was developed by Microsoft through a sequence-to-sequence framework that includes encoder-decoder components [50]. It applies a self-attention mechanism to each section of the encoder and decoder to determine word contextual bonds and produces precise translations for multiple language pairs.

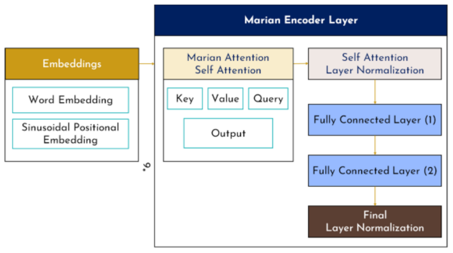

Marian Encoder Layer
Embeddings
Word Embedding
Sinusoidal Positional Embedding
Marian Attention Self Attention
Key
Value
Query
Output
Self Attention Layer Normalization
Fully Connected Layer (1)
Fully Connected Layer (2)
Final Layer Normalization
6

Fig. 9. Marian encoder architecture [51]

.

MarianMT displays the capability to achieve precise English-to-ASL gloss mapping through model training in ASL gloss conversion scenarios. A unique ability to normalize linguistic patterns and preserve text data quality during preprocessing positions it as an ideal solution for ASL gloss sequence production. Due to its multilingual character, MarianMT optimizes transitions between various grammatical frameworks that exist in ASL.

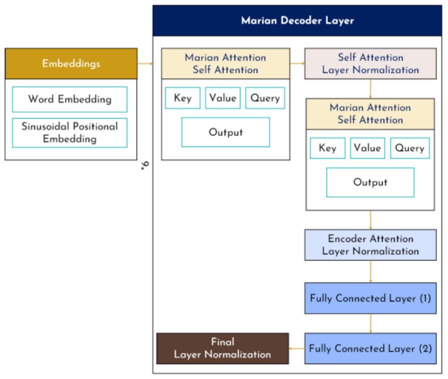

Marian Decoder Layer
Embeddings
Word Embedding
Sinusoidal Positional Embedding
Marian Attention Self Attention
Key
Value
Query
Output
Self Attention Layer Normalization
Marian Attention Self Attention
Key
Value
Query
Output
Encoder Attention Layer Normalization
Fully Connected Layer (1)
Final Layer Normalization
Fully Connected Layer (2)
6

Fig. 10. Marian decoder architecture [51]

.

MarianMT achieves efficiency along with a compact design due to its ability to integrate its complete translation workflow directly into its neural network architecture. The vocabulary compression techniques within MarianMT employ subword tokenization to minimize model size while maintaining an excellent performance level suitable for real-time gloss generation needs in limited resource situations.

C. Mapping Text to 3D Keypoints

After converting English text to ASL gloss, the next step is to map the gloss tokens to corresponding 3D key points. These key points represent anatomical landmarks such as finger joints, hand positions, and body posture, essential for generating realistic ASL gestures. The processes are below:

1) **Dataset Preparation:** We obtain 3D key points from video datasets through the implementation of models including OpenPifPaf, ResNet50, RTMPose3D, and MediaPipe. We extract 3D key points from video datasets. MediaPipe captures 21 key points for each hand. OpenPifPaf helps with 2D human pose estimation and provides precise joint locations. ResNet50 is used for feature extraction to recognize complex patterns within these 3D key points. RTMPose3D enhances the datasets by offering higher precision in joint coordinates.
2) **Mapping and Matching:** The system links gloss tokens to the existing key points that were previously added to the dataset. A similarity-matching algorithm tracks gloss tokens to identify their closest motion gesture relation for solving the alignment issue between writing gestures and text output.

D. Hand Gesture Modeling

To get gesture accuracy, we integrate 2D as well as 3D hand gesture models as an intermediate step. These models refine the representation of hand movements before transitioning to 3D animation. The model descriptions are below:

1) OpenPifPaf + JointFormer Dataset: The OpenPifPaf model was developed by Kreiss et al. (2021) [52] which delivers high-precision detection of the human body and hand joints. Through part-intensity fields, the model locates key points throughout images by analyzing spatial body joint connections. Through this ability, OpenPifPaf provides efficient and precise real-time pose analysis so users can benefit from gesture detection. Its main advantage includes precise modeling of anatomical interactions between different joints.

The efficiency of OpenPifPaf comes from its lightweight architecture, which allows it to be deployed in real-time applications without significant computational overhead. A key benefit of OpenPifPaf is its ability to identify key points from image or continuous video frames thus advancing rapid dependable ASL gesture recognition. Experimental evidence shows OpenPifPaf successfully works with part-intensity fields when occlusion occurs and body parts are partially hidden in the scene. Experimental evidence shows OpenPifPaf successfully works with part-intensity fields when occlusion occurs and body parts are partially hidden in the scene. According to Kreiss et al. (2021) [52], OpenPifPaf performs particularly well in real-life situations with moving hands because it handles hand blockage and limits overlapping between hands.

The framework JointFormer enhances OpenPifPaf through its addition of temporal consistency mechanisms between

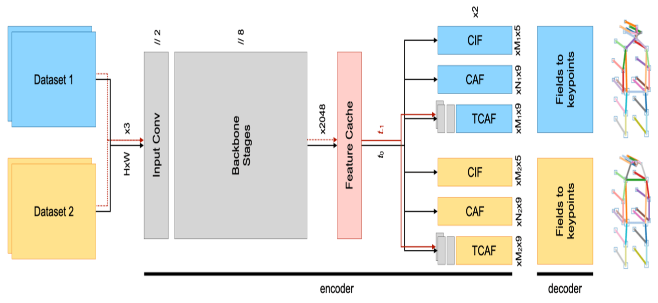


Fig. 11. OpenPifPaf Model Architecture [52]

.

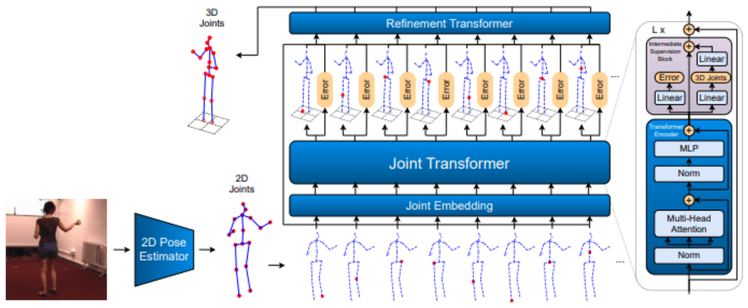


Fig. 12. JointFormer for single-frame 2D-3D human pose lifting [53]

.

consecutive frames. The tracking ability of transformer-based attention approaches helps attain this goal by following key points' movements across frames. Through this system detected hand gestures create clean uninterrupted motion channels vital for modeling how ASL users move their hands dynamically. The merged OpenPifPaf and JointFormer system maintains the second-best quality gesture tracking by providing comprehensive spatial and temporal pose estimation capabilities [53].

2) ResNet50: The ResNet50 is a deep convolutional neural network which Microsoft Research designed to use residual learning to train deep network architectures while solving the gradient vanishing phenomenon [54]. Residual learning capabilities are introduced through identity shortcuts which allow the model to understand residual transformations rather than traditional direct mappings. Such a design structure maintains its importance for extracting essential features from complex hand gestures because deep hierarchical representations are essential. The mathematical expression for a residual block is given by:

$$H(x) = F(x, \{W_i\}) + x$$

where:

- $H(x)$ is the output,
- $F(x, \{W_i\})$ represents the residual mapping, and
- x is the input.

ResNet50 becomes a preferred choice in visual recognition tasks because it adequately extracts low intermediate and high-level features from images in visual detection tasks like facial expression and hand gesture analysis. The ability to interpret detailed poses along with context-based image relationships defines the sign language recognition performance of the network system. The network design enables organized feature extraction which remains accurate during complex gestural movements that feature complex hand directions or camera coverages

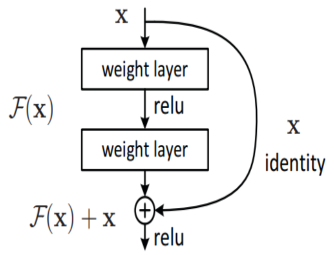


Fig. 13. Residual learning: a building block [54]

.

The ResNet50 model processes data through 48 convolutional layers in sequence. Each layer is followed by batch normalization and ReLU activation. Residual blocks in the network allow gradients to skip over several layers so deeper networks achieve fast convergence.

3) RTMPose3D: RTMPose3D builds on the RTMPose framework, which is designed for real-time multi-person pose estimation and enhances its functionality to the 3D domain. RTMPose features a swift architecture that runs on lightweight components which deliver accurate and efficient performance well suited for ASL gesture detection systems [55]. The model supports Convolutional Neural Networks (CNNs) for spatial feature identification at low latency rates.

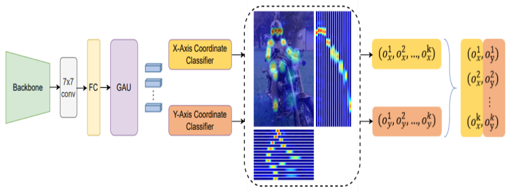

Backbone
7x7 conv
FC
GAU
X-Axis Coordinate Classifier
Y-Axis Coordinate Classifier
$(o_x^1, o_x^2, ..., o_x^k)$
$(o_y^1, o_y^2, ..., o_y^k)$
(o_x^1, o_y^1)
(o_x^2, o_y^2)
(o_x^k, o_y^k)

Fig. 14. The overall architecture of RTMPose [55]

.

RTMPose3D developed the core RTMPose design foundation while adding 3D pose estimation functionality. The system efficiently detects key points regardless of body types or positions through Feature Pyramid Network(FPN) optimizations combined with additional performance enhancements. RTMPose3D attains accurate 3D key point positioning through the integration of multi-view frameworks with triangulation tactics.

RTMPose3D delivers real-time feature analysis, enabling effective live ASL interpretation in resource-limited systems. Temporal smoothing features, along with other methods, create a system that reduces flickering and tracks gesture movements more smoothly while protecting ASL communication fluidity.

4) MediaPipe: MediaPipe is an open-source framework developed by Google which enables users to create multi-modal perception pipelines. The framework offers real-time performance metrics while further focusing on hand-detection capabilities and gesture recognition software according to Lugaresi et al. (2019) [56]. The MediaPipe framework uses MobileNet architecture and features from the MobileNet family to track 21 distinctive points within each hand where fingers meet and arms and hands connect. This framework is both efficient and flexible. It can run successfully on a variety of platforms. These include desktop systems and resource-limited edge devices, such as smartphones.

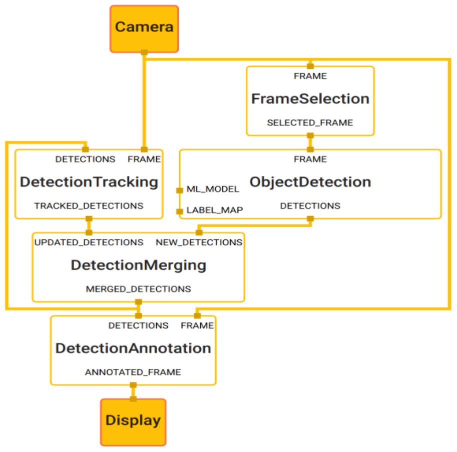


Fig. 15. Object detection using MediaPipe [56]

.

Through its pipeline-based design, MediaPipe enables real-time tracking of continuous video streams. MediaPipe operates with two stages: detection and tracking. This process involves detecting and identifying hands within images, followed by frame-by-frame monitoring of hand movements to track their dynamic actions accurately. MediaPipe uses its tracking system to deliver accurate results. It ensures fluid transitions, which are essential for interpreting dynamic hand gestures in ASL.

Through its custom input and output configuration features the framework gives developers substantial control to integrate models and pipelines. MediaPipe demonstrates excellent computing power and wide platform compatibility that enables efficient real-time gesture detection within constrained computational environments [56].

E. Final Animation Output

This part explains how to create animations by utilizing the data resources and computational models used throughout this study. The methodology brings together raw data from the How2Sign database [57] alongside computational outputs derived from key point extraction algorithms stemming from the ”Hand Gesture Modeling” pipeline. The long short-term memory LSTM model serves as the backbone for middle frame generation implementation.

1) Animation Workflow: The How2Sign dataset provides the foundation for modeling continuous American Sign Language (ASL). The How2Sign dataset represents one of the largest multimodal datasets that contains synchronized video as well as, depth information and skeleton spatial data for ASL [57]. Various gesture recognition processes alongside translation and synthesis run tests against this benchmark. The dataset serves as the basis for creating temporal data that enables the production of smooth animated ASL gestures.

The animation generation process receives input through the keys extracted by the four modeling approaches (OpenPifPaf + JointFormers, ResNet50, RTMPose3D, and Mediapipe) in addition to the How2Sign data collection. The skeletal movement information needed for signing appears within these data points. A smooth gesture transition can be achieved through the LSTM model which predicts and generates middle frames during shifts between gestures. An animation of smooth transitions emerges when predicted frames receive key point data for their synchronization process. These animated outputs unify into one aesthetically consistent visual presentation of ASL.

2) LSTM Model for Animation Generation: LSTM or Long Short-Term Memory model is a special type of RNN(Recurrent neural network) which can analyze sequential data effectively by maintaining long-term dependencies. The LSTM model was first introduced by Hochreiter and Schmidhuber (1997) [58] which incorporate memory cells with gating controls that make them adaptable to both immediate and distant relationships in sequential data streams. In this research, the LSTM is used to generate intermediate key points which are also known as middle frames, between consecutive gestures. The implementation of middle frame generation in animations maintains stable transitions from gesture to gesture through smooth natural animation movements by addressing immediate motion variations.

The middle frame generation process involves training the LSTM using sequential key points derived from the How2Sign dataset and the outputs of the four key point modeling frameworks: OpenPifPaf + JointFormers, ResNet50, RTMPose3D, and Mediapipe. The LSTM framework analyzes gesture motion by understanding temporal connections between key point movements for automatic skeletal point prediction during gesture transitions. Animation outputs become more realistic because this method completes the transmission between various gesture sequences. The mechanism plays an essential role since it eliminates the discontinuous and rapid visual flow which characterizes many animations without temporal buffering.

Middle frame generation with LSTM draws its foundation from recent developments in motion synthesis as well as temporal modeling techniques. Fragkiadaki et al. (2015) [59] achieved significant progress with LSTM-based networks for human motion prediction and established their capability to model diverse temporal behavioral changes. Similarly, the LSTM in this research collects temporal information from How2Sign to match generated intermediate frames with natural ASL temporal patterns. The design technique makes sure every motion transition merges finely which produces animations with natural movement qualities. The addition of middle frames as an animation quality enhancement further builds the accessibility of American Sign Language throughout virtual environments for diverse audiences.

VI. Implementation

A. *Workflow for Translating Speech to 3D ASL Representation*

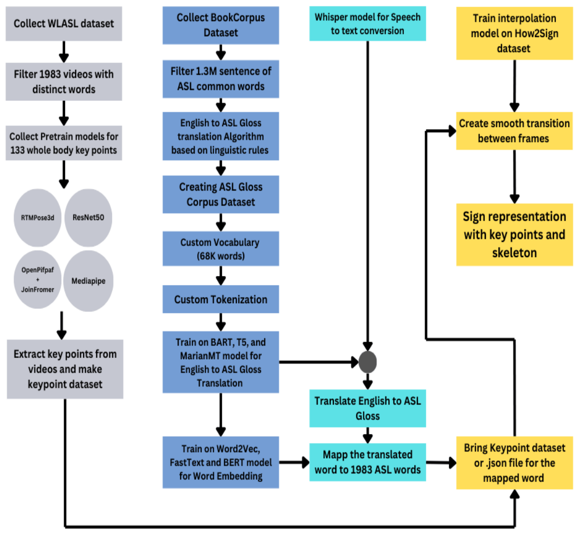

Collect WLASL dataset
Filter 1983 videos with distinct words
Collect Pretrain models for 133 whole body key points
RTMPose3d
ResNet50
OpenPifpaf + JoinFromer
Mediapipe
Extract key points from videos and make keypoint dataset
Collect BookCorpus Dataset
Filter 1.3M sentence of ASL common words
English to ASL Gloss translation Algorithm based on linguistic rules
Creating ASL Gloss Corpus Dataset
Custom Vocabulary (68K words)
Custom Tokenization
Train on BART, T5, and MarianMT model for English to ASL Gloss Translation
Train on Word2Vec, FastText and BERT model for Word Embedding
Whisper model for Speech to text conversion
Translate English to ASL Gloss
Mapp the translated word to 1983 ASL words
Train interpolation model on How2Sign dataset
Create smooth transition between frames
Sign representation with key points and skeleton
Bring Keypoint dataset or .json file for the mapped word

Fig. 16. Workflow Diagram

Translating speech into 3D ASL representation is combined with multi-step. Each section of the workflow operates to achieve maximum accuracy and operational effectiveness to produce final smooth ASL translations. The initial stage of ASL translation requires the collection and enhancement of data sets needed for processing. The WLASL database consists of 1,983 ASL word videos that undergo filtering processes to keep only unique distinct word videos. Complex sign movements can be extracted through the application of four pretrained models including RTMPose3D [55], ResNet50 [54], OpenPifPaf [52] + JoinFromer [53], and Mediapipe [56] to capture complete body key points. The key point extraction capabilities of these models analyze ASL videos through 133 identifiable points that later facilitate 3D representation production.

The second step, Text Processing and ASL Gloss Translation which converts written text into ASL gloss that acts as an intermediate step for ASL translation. BookCorpus [60] dataset collection starts with filtering 1.3 million sentences which contain common ASL words to finalize the implementation of a rule-based English-to-ASL gloss translation algorithm. The processed BookCorpus data is transformed into an ASL Gloss Corpus with support from an organized vocabulary of 68,000 words. A specific approach for marking the input text into gloss format has been developed to enhance ASL gloss translation. Automatic translated glosses for English to ASL text are possible through BART [42] and MarianMT [50] while Word2Vec [44] [45] serves alongside FastText and BERT [61] for word embedding and contextual meaning improvement.

The third step is to convert Speech to ASL Gloss which requires the transformation of spoken language into ASL gloss which must be compared to previously added ASL words. The Whisper model [41] provides speech-to-text conversion to generate accurate English transcription from spoken words. After text translation, the system conducts processing through the trained ASL gloss model to link incoming text content with the 1,983 ASL words that have been extracted previously. Each sentence input receives an ASL representation during this process that establishes the base for sign animation development.

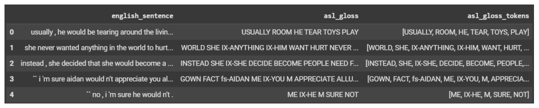
Finally, 3D Sign Representation and Animation completes the process by creating real-life animated ASL signs. For a smooth transition, we used the interpolation model and we used the How2Sign dataset [57] to train that interpolation model. The collected key points enable the creation of skeletal representations that depict ASL signs. The animation of the precise 3D ASL word representation is achieved by employing the keypoint dataset or JSON file.

B. Dataset Pre-processing

A proper dataset preparation system is essential for achieving accurate and realistic performance in ASL translation systems. Our system requires multiple datasets which support the translation from speech to 3D ASL representations by addressing separate steps in the translation process. The study relies on three essential datasets including the Word Corpus to ASL Gloss Corpus Dataset, WLASL to Keypoint Dataset, and How2Sign Dataset. The datasets undergo thorough preparation to support two separate models that translate texts into signed expressions and generate 3D motions. The combination of datasets allows the system to achieve complete control over the linguistic together with a motion-based representation of ASL signs.

C. Word Corpus to ASL Gloss Corpus Dataset

The Word Corpus to ASL Gloss Corpus Dataset offers the base for text-processing along with providing precise ASL gloss translations from English text. We developed the dataset from the BookCorpus dataset while gathering an extensive variety of written texts. A separate grammatical transformation process operates on ASL texts because this sign language differs from English linguistic rules. The text selection process starts with choosing 10,000 of the most common ASL words to obtain data that represents authentic ASL communication. Then we expanded the dataset to contain 1.3 million sentences by finding the optimal value for computational efficiency.



	english_sentence	asl_gloss	asl_gloss_tokens
0	usually , he would be tearing around the livin...	USUALLY ROOM HE TEAR TOYS PLAY	[USUALLY, ROOM, HE, TEAR, TOYS, PLAY]
1	she never wanted anything in the world to hurt...	WORLD SHE IX-ANYTHING IX-HIM WANT HURT NEVER ...	[WORLD, SHE, IX-ANYTHING, IX-HIM, WANT, HURT, ...
2	instead , she decided that she would become a ...	INSTEAD SHE IX-SHE DECIDE BECOME PEOPLE NEED F...	[INSTEAD, SHE, IX-SHE, DECIDE, BECOME, PEOPLE,...
3	`` i 'm sure aidan would n't appreciate you al...	GOWN FACT fs-AIDAN ME IX-YOU M APPRECIATE ALLU...	[GOWN, FACT, fs-AIDAN, ME, IX-YOU, M, APPRECIA...
4	`` no , i 'm sure he would n't .	ME IX-HE M SURE NOT	[ME, IX-HE, M, SURE, NOT]

Fig. 17. BookCorpus Dataset to ASL Gloss Corpus Dataset translation

.

The conversion process of English text to ASL language-suitable formats is supported by a rule-based ASL gloss translation algorithm. The algorithm used restructuring operations on ASL linguistic conventions through a rule-based transformation model. Using the spaCy NLP library enables the implementation of transformation processes that analyze sentence structures. The methodology uses word classification according to syntactic roles to identify temporal words wh-words and fingerspelled lexical words as part of appropriate ASL grammar representation.

The algorithm operates as follows:

1) The sentence analysis occurs through the `en_core_web_sm` model of spaCy on English text.
2) The system integrates a component extraction procedure for subjects and verbs and adjectives then processes their rearrangement based on American Sign Language grammar traditions.
3) The algorithm processes negations wh-questions and temporal expressions to preserve linguistic correctness of ASL gloss.
4) The algorithm divides complex sentences with compound-complex sentence parsing that enforces ASL grammatical rules for each segment.

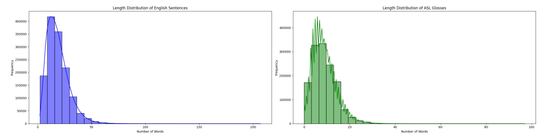


Fig. 18. fig:Frequency vs. Word Length for English and ASL words

.

The dataset contains ASL gloss translations together with tokenized versions that will serve as training material for deep learning models. This final transformation process helps the text-to-ASL gloss model receive direct training through ASL glosses which follow authentic ASL grammatical patterns.

D. WLASL to Keypoint Dataset

The 3D hand sign generation model receives essential training data from the WLASL to Keypoint Dataset which contains real ASL gesture information. The WLASL dataset includes 1,983 high-quality videos showing defined and articulated ASL sign words after strict selection criteria. The analysis process involves a frame-by-frame examination to obtain 133 key points which primarily monitor essential upper body movements together with hand gestures required for sign language contents. The keypoint extraction process becomes more resilient through employing several deep learning models that solve different aspects of pose estimation problems.

1) RTMPose3D for Keypoint Extraction: The RTMPose3D [55] method retrieves 133 3D key points from images and videos through its training with COCO WholeBody dataset information. RTMDet serves as the first stage to detect human subjects before RTMPose3D implements 3D point estimation. Before the pose estimation process begins the system performs data pre-processing operations on video frames to make them suitable for evaluation. The RTMPose3D pipeline performs model training on each frame to create JSON files that contain predicted 3D coordinate values for the nose and several body area points such as shoulders and elbows, wrists, and fingers. The processed information contains detailed hand joint positioning needed for sign language interpretation. The full-body pose estimation function of RTMPose3D tracks hand arm and facial movements precisely which ASL interpretation strongly depends on.

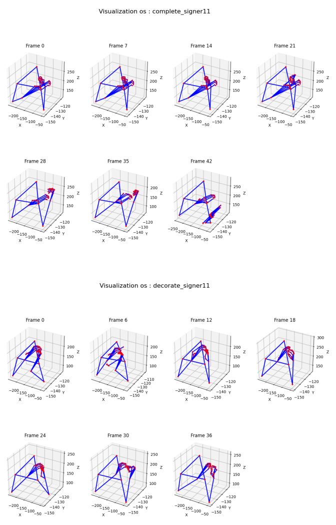


Fig. 19. RTMPose3D 3D Skeleton Visualization

2) ResNet50-Based Pose Estimation: The key point extraction system uses ResNet50 [54] features to extract information from human pose data during human pose estimation operations. The framework undergoes training on large datasets and obtains specialized competence for human body structure detection by utilizing the H3WB dataset for upper-body and hand identification. During the execution of convolutional layers multiple times images produce deep feature extracts which represent body parts. The refined features proceed to an extra regression module which predicts both two-dimensional and three-dimensional points of key locations. The implementation uses ResNet50 to collect high-level spatial information from input frames which becomes essential for joint position and hand movement identification. Above all the system connects specialized expertise to track complex signs from American Sign Language accurately.

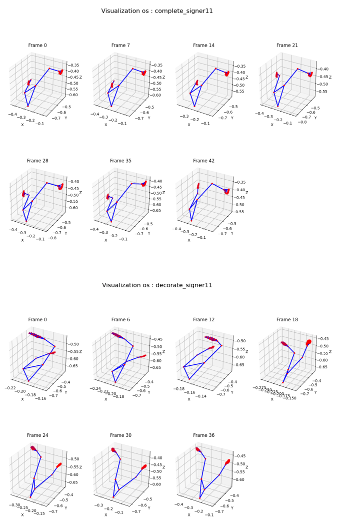


Fig. 20. ResNet50 3D Skeleton Visualization

3) OpenPifPaf + JointFormer for Pose Estimation: The OpenPifPaf [52] model works alongside JointFormer [53] to identify human body points with exceptional precision along with abilities to analyze multiple individuals in the scene. OpenPifPaf operates as a bottom-up model which detects body segments before linking them into complete body poses. OpenPifPaf extracts 2D key points that JointFormer receives as input to improve the pose accuracy through spatial keypoint relationship refinement. By combining COCO WholeBody and H3WB dataset information the system improves keypoint detection which leads to more reliable motion prediction for American Sign Language representation. The video processing consists of OpenPifPaf keypoint detection followed by JointFormer which strengthens keypoint detection consistency and robustness to provide tight control of hand movement tracking for sign language transcription purposes.

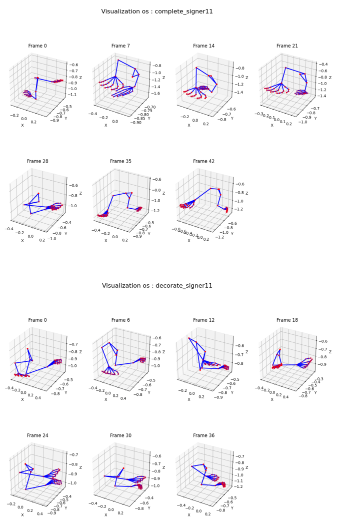


Fig. 21. OpenPifPaf + JointFormer 3D Skeleton Visualization

4) MediaPipe for Real-Time Keypoint Extraction: MediaPipe [56] model extracts key points from videos in real-time using a lightweight machine learning pipeline. The model detects complete poses through body and face and hand key point identification. A pose detector identifies people first and after that, the landmark model provides precise joint position estimates. The extraction process of MediaPipe combines COCO WholeBody data with a Lifting Dataset to achieve 3D reconstruction. A consistent depth together with the spatial arrangement of key points results in enhanced accuracy when representing ASL signs in 3D. Hand joints along with other extracted key points get processed to identify individual finger movements because these movements are vital for recognizing sign language. The real-time performance of MediaPipe tracking operates through an efficient system that works with small computational needs suitable for live pose estimation applications.

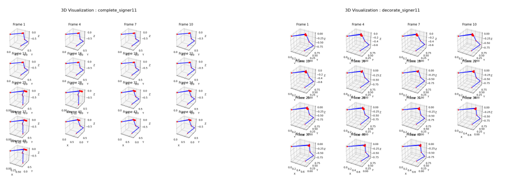


Fig. 22. MediaPipe 3D Skeleton Visualization

By integrating these models, the system generates precise 3D representations of ASL gestures which ensure accurate animations for sign language communication.

E. How2Sign Dataset

The researchers use the How2Sign dataset [57] to build better 3D animated ASL gestures. We use the MediaPipe [56] model to analyze the dataset containing synchronized ASL video transcriptions by extracting 47 upper-body key points for each frame that detect hand, arm, and facial expressions. A Bi-LSTM model accepts key points obtained from their extraction along with structured sequences for processing. The Bi-LSTM model employs sequential data handling excellence through its forward-backward direction learning approach to guarantee uninterrupted key point predictions between American Sign Language gestures. JSON files contain structured data from the extraction while MinMaxScaler normalization protects consistency between different signing individuals and video inputs.

A `process_video()` function extracts key points per frame before `find_sequences()` arranges collected data into sequences. The sequences get processed for training the Bi-LSTM model and must contain X frames as input data, while Y contains generated intermediate frames which maintain smooth motion transitions. A 3D environment benefits from this technique because it produces sign language expressions that are more natural and vibrant.

F.

sectionModel Training and Evaluation

G. Whisper Model Training

Text conversion from speech forms the base component in our translation system because it delivers input data for the conversion of ASL glosses. Our system employs the Whisper model [41] that is trained through 680,000 hours of multilingual data which makes it work properly for various speech styles and environments. The pre-trained model gives us high transcription accuracy without requiring extensive retraining. Our ASL gloss conversion operation benefits from text normalization methods which remove punctuation while filtering stop words and dividing the text into segments. Text preprocessing for the ASL language integrates it into ASL grammar rules before moving forward to the translation stages.

H. ASL Gloss Translation Model Training

The three translation model types including BART, MarianMT, and T5 received initial training to become functional. The training dataset contained 68,000 ASL words so they underwent tokenization and ID conversion to be suitable for all three models. Training policies directed the handling of out-of-vocabulary (OOV) tokens during the process to convert words beyond the knowledge base into placeholders without causing disruptions.

The following characteristics defined the training procedures for the models. For the BART model use use 1 epoch, for the MarianMT and T5 models, we 5 epochs each. BART and MarianMT processed their batches using a capacity of 128 while T5 operated with a capacity of 256. Learning rate decay enhanced convergence while the models received optimization through the Adam optimizer. Basic cross-entropy loss function helped minimize translation mistakes during training.

These models showed continuous positive changes in their training loss curves during their training process. The table below compares the training and validation loss of BART, MarianMT, and T5 at different training steps. It highlights each model's learning progression and convergence efficiency.

Step	**BART**		**MarianMT**		**T5**	
	Train Loss	**Val Loss**	**Train Loss**	**Val Loss**	**Train Loss**	**Val Loss**
2000	2.0584	0.5435	0.0939	0.0869	2.3337	2.1275
4000	0.4468	0.2921	0.0944	0.0821	2.2124	2.0184
6000	0.2922	0.2213	0.0879	0.0784	2.0599	1.9418
8000	0.2396	0.1960	0.0707	0.0757	2.0599	1.9079
10000	–	–	0.0672	0.0735	2.0254	1.8824
12000	–	–	0.0656	0.0718	2.0051	1.8766

TABLE I
TRAINING AND VALIDATION LOSS FOR BART, MARIANMT, AND T5 MODELS

BART Loss Curve: The model shows a rapid decrease in both training and validation loss from 2000 to 8000 steps, indicating good convergence. However, loss values beyond 8000 steps are missing because we only run one epoch, making it unclear whether further improvement occurs.

MarianMT Loss Curve: This model shows the lowest and most stable loss values throughout training, with validation loss reducing from 0.0869 at 2000 steps to 0.0718 at 12000 steps. It demonstrates the best convergence and generalization among the three models.

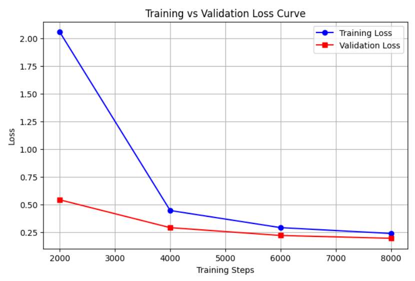


Fig. 23. BART Loss Curve

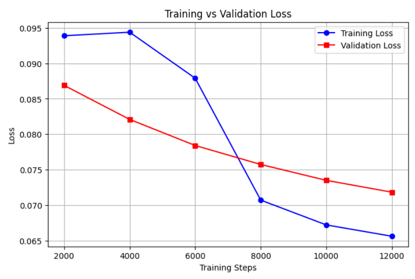


Fig. 24. marianMT Loss Curve

T5 Loss Curve: While initially having higher loss values, T5 shows a gradual decline in both training and validation loss from 2000 to 12000 steps. Despite this, its final loss values (1.8824 validation loss) remain higher than MarianMT, suggesting slower convergence and potentially weaker generalization.

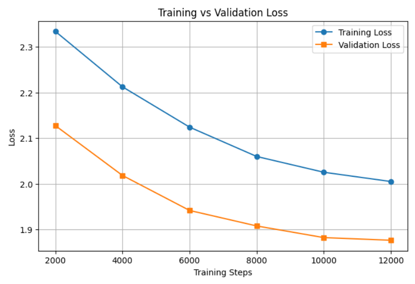


Fig. 25. T5 Loss Curve

I. Keypoint Extraction Model Training

Translation of American Sign Language into 3D animations heavily depends on the key point extraction process. Training with frames from the WLASL dataset retrieved hand and upper-body key point information that we normalized for maintaining stability across miscellaneous body sizes and cameral perspectives.

The keypoint extraction process incorporated four models and they are RTMPose3D, ResNet50, MediaPipe, and OpenPifPaf + JointFormer. The models received training through the Colab T4 GPU environment due to its high processing power that optimized training efficiency. The open PifPaf platform with JointFormer gained popularity because it delivers excellent results when detecting body points while processing sequences through time.

OpenPifPaf learned new parameters from the WLASL dataset to identify body key points precisely. The temporal connections between key point positions needed to be detected and processed by JointFormer within video sequences for creating accurate 3D representations of moving ASL gestures. The cross-entropy loss served as the training objective to reduce key point positioning mistakes.

J. 3D Sign Animation Model Training

For the Linear Interpolation model, we computed the average of key points between two frames to generate intermediate frames, filling gaps between keyframes. Using these interpolation models, we generated missing frames between each keyframe. To train the interpolation models, we used an 80:10:10 train-validation-test split. After training, we visualized the key point skeletons using the Matplotlib library, where key points for each frame were plotted sequentially to generate smooth animations.

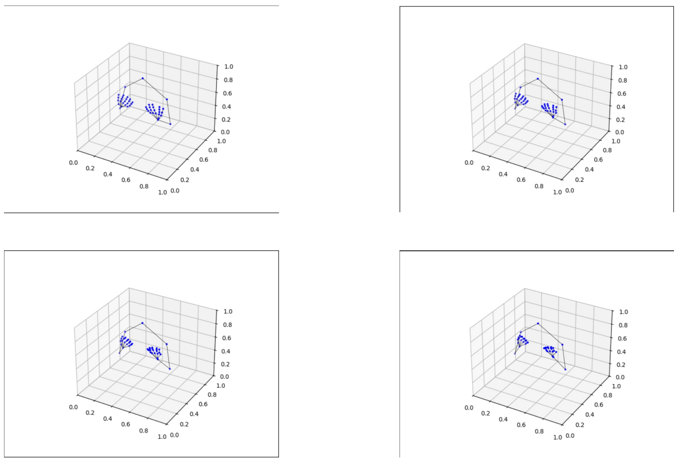


Fig. 26. Linear Interpolation Model Frames

The Bi-LSTM model worked as a keyframe interstitial system that generated smooth animation transitions for 3D ASL sign technologies. The How2Sign dataset achieved 80-20 distribution for training purposes and test purposes following typical machine learning model practices. The continuous sign language videos and their associated 3D pose annotations in the dataset supply extensive motion sequences for learning. The model obtained better generalization by incorporating data augmentation methods which included frame drops in random order combined with key point movement changes and temporal length adjustments. We trained the model with sequence-to-sequence loss optimization with Adam optimizer while the learning rate adapted automatically using validation results.

During the training process, the Bi-LSTM model's loss decreased steadily and stabilized after 10000 iterations which showed it had learned successfully. The model proved able to generalize effectively on new sequences because test loss values stayed low. The shape constraints functioned to stop unnat-

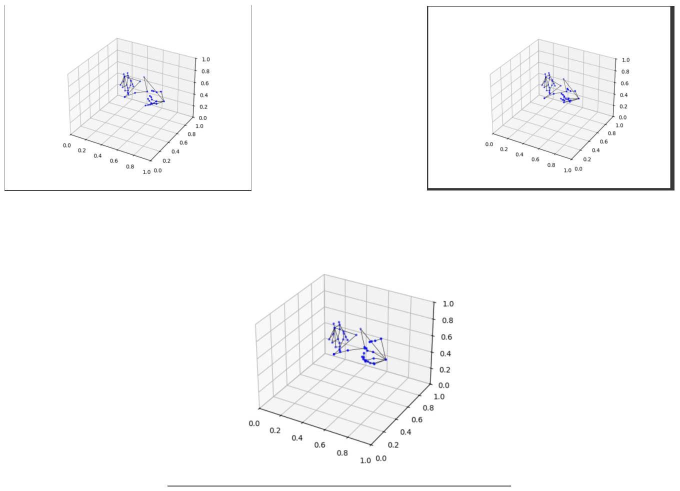


Fig. 27. Actual Frames Collected from Video

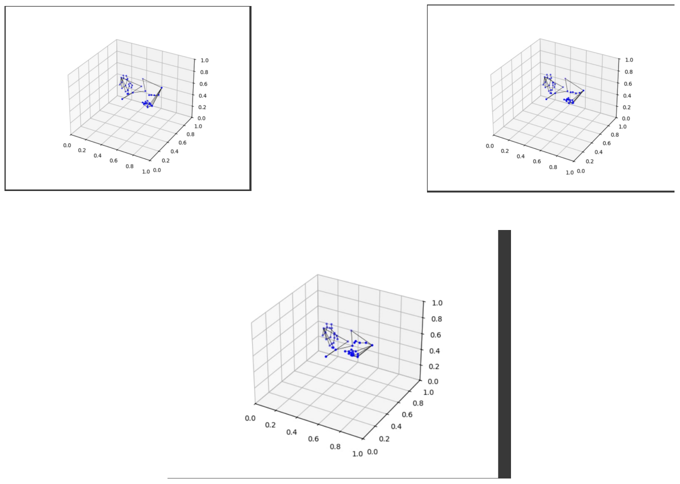


Fig. 28. Bi-LSTM Predicted Frame

ural body deformations by maintaining both joint angles and proportional relationships within natural human limitations. The smooth transitions from Bi-LSTM were inferior to linear interpolation because Bi-LSTM could not track motion dependencies. The naturality of sequences produced by Bi-LSTM was verified through velocity and acceleration measurements between frames and expert evaluations demonstrated proper representation of ASL gestures. The combination of machine learning-based interpolation techniques shape constraints and linear interpolation constructs refined the fluency alongside accuracy and anatomical correctness in the generated 3D ASL animations.

VII. Result and Comparison

A. BLEU Score Analysis

We utilized the well-known Bilingual Evaluation Understudy (BLEU) [62] score for measuring text translation accuracy into American Sign Language (ASL) within our model. BLEU [62] evaluates machine translation outputs by investigating the parallelism between model-produced sentences and reference translations whereas it deducts points from translations that lack sufficient length. The measurement scale goes from 0 to 1 and provides a rating of translation quality based on its values. The formula for the BLEU score is as follows:

$$BLEU = BP \times \exp\left(\sum_{n=1}^{N} w_n \cdot \log P_n\right) \quad [63]$$

Where:

- BP is the brevity penalty, which penalizes translations that are shorter than the reference translations.
- P_n is the precision for each n-gram level (from unigram to N-gram).
- w_n is the weight assigned to each n-gram precision (commonly, each w_n is set to $\frac{1}{N}$).
- N is the maximum n-gram order, typically 4 (i.e., unigrams, bigrams, trigrams, and 4-grams).

Components of BLEU Calculation: **Precision of n-grams (P_n):** Precision for a particular n-gram length accounts for the number of machine translation n-grams which correspond to those in reference translations. The precision measure for each n-gram level from 1 to N can be calculated through this method:

$$P_n = \frac{\text{Number of matching n-grams}}{\text{Total number of n-grams in the machine translation}} \quad [63]$$

Brevity Penalty (BP): The brevity penalty ensures that shorter translations are penalized. It is calculated as:

$$BP = \begin{cases} 1 \\ \exp\left(1 - \frac{\text{reference length}}{\text{translation length}}\right) \end{cases} \quad [63]$$

1 applies when the translation length reference length, and $\exp\left(1 - \frac{\text{reference length}}{\text{translation length}}\right)$ applies when the translation length $<$ reference length.

Here, the term ”translation length” measures the number of words within machine output and ”reference length” indicates the word count of reference translations. [63].

Final BLEU Score:

$$BLEU = BP \times \exp\left(\sum_{n=1}^{N} \frac{1}{N} \log P_n\right) \quad [63]$$

The BLEU score calculation uses the average of n-gram precision values taken from their logarithmic summation (all n-gram orders have equal weighting). A BLEU score final calculation considers both the matching n-grams and the translation length in relation to the reference text. [63].

We tested three models T5, BART, and MarianMT and evaluated their performance at both the sentence level (how well individual sentences match the reference) and the corpus level (how well the model performs across the entire dataset). Here’s how they performed:

The MarianMT model produces better results than both T5 and BART models according to both sentence-based and

Model	Average Sentence-Level	Corpus-Level
T5 Model	0.0184	0.0634
BART Model	0.5577	0.7067
MarianMT Model	0.7714	0.8923

TABLE II
TABLE 5.1: BLEU SCORE COMPARISON OF MODELS

overall BLEU score measurements. MarianMT showcases the peak scores of 0.7714 for average sentence level BLEU and 0.8923 for the corpus-level BLEU score. The T5 model produces notably lower outcomes since it attains 0.0184 as its sentence-level BLEU score and 0.0634 as its corpus-level BLEU score. BART demonstrates superior performance to T5 according to BLEU scoring at 0.5577 sentences and 0.7067 corpora. Current research shows that MarianMT demonstrates maximum effectiveness as an English to ASL gloss conversion model based on both statement-level and collection-level translation quality evaluation. Additional research involving investigations into MarianMT's superiority would result in enhanced understanding of its core operational methods.

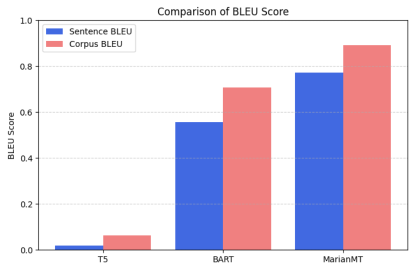


Fig. 29. Comparison of BLEU Scores Across Different Models for English to ASL Gloss Conversion

A BLEU score of 0.8923 suggests that the MarianMT model's translations are very close to human-generated ASL representations.

B. Interpolation Result Analysis

As our dataset consists of synthesized sign language videos, the accuracy of 3D key point detection was inherently affected by errors in coordinate extraction. Inaccuracies in elements affected how well the interpolation strategies functioned since they reduced the quality of frame restoration methods. Linear interpolation outperformed other interpolation approaches because it maintained smooth motion alongside high gesture quality. The effectiveness of this method depends on having accurate key sample annotations but real-world conditions make it difficult to obtain these annotations precisely. Natural motion transitions during American Sign Language signing become difficult to capture through linear interpolation if keyframe annotations lack accuracy. The linear interpolation formula implemented in our method functions through the following definition:

Formula for Linear Interpolation:

$$\text{frame}_a[i] = \frac{\text{frame}_{a-x}[i] + \text{frame}_{a+x}[i]}{2}$$

where a is the number of the desired frame, i denotes the coordinate to be calculated, and x can be any number such that both the $(a-x)$th and $(a+x)$th frames exist.

The Bi-LSTM interpolation approach brought promising solutions to the table though it experienced several system constraints. The model created suitable bridge connections between frames yet misplaced some important points resulting in small visual disruptions in the final video generation. The model sometimes produced jittering artifacts which can be attributed to prediction inconsistencies it displayed between sequential frames. The Bi-LSTM demonstrates solid temporal dependency modeling however proper customization or subsequent processing methods should be employed to improve both steadiness and accuracy during sign language motion synthesis tasks.

The loss function assessment included an examination of training loss performance between both interpolation techniques. MPJPE (Mean Per Joint Position Error) is used as the loss function. It measures the Euclidean distance of the actual and predicted coordinate for each frame and then calculate the mean. The lower the value, the higher the accuracy of the predicted frames. The Bi-LSTM model achieved a training loss of 0.04461418464779854 during the training phase while linear interpolation reached 0.04297982136398096 as its loss. Even with its ability to learn sequence dependencies the Bi-LSTM model needs additional optimization to produce continuous movement transitions. Linear interpolation remains effective when sufficient high-quality keyframe annotations are used regardless of the small training loss difference.

Model	MPJPE Score
LSTM	0.044
Linear Interpolation	0.0429

TABLE III
COMPARISON OF MPJPE SCORES FOR DIFFERENT MODELS

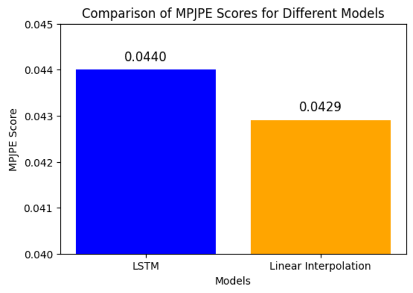


Fig. 30. MPJPE Scores graphical representation

To evaluate the effectiveness of different models for 3D human pose estimation, we compare their MPJPE scores. A lower MPJPE value indicates a more accurate prediction of joint positions, thereby reflecting better performance.

Lastly, we see that the Linear Interpolation model has a slightly lower MPJPE score (0.0429) compared to the LSTM

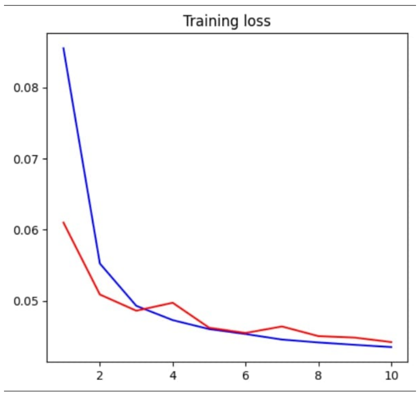


Fig. 31. Bi-LSTM Loss Curve During Training

model (0.044) from the below graph. Since MPJPE measures the average error in joint position estimation, a lower value indicates better accuracy. Therefore, in this comparison, the Linear Interpolation model outperforms the LSTM model in terms of raw MPJPE.

Text generation with Bi-LSTM produces better results because it learns sequential relationships between frames but linear interpolation depends solely on correct keyframes. Evidence suggests that Bi-LSTM can achieve better accuracy since it resulted in a final loss value of 0.019991353154182434. Linear interpolation demonstrates acceptable performance based on these results while maintaining simplicity as well as efficiency in computations. A motion-captured dataset substitution for synthesized data would boost interpolation performance especially when using Bi-LSTM technology according to experimental findings. Motion-captured data provides accurate keypoint annotations which decreases coordinate detection errors and results in better fluid motion of sign animation interpolation outputs.

VIII. Conclusion

A. Future Improvements

The future improvement of this work should emphasize enhancing the training methods which power sign language interpolation along with translation mechanisms. The learning process can be enhanced by developing better hyperparameter set-ups together with data augmentation methods along with sophisticated loss functions to achieve improved accuracy and robustness across models. The English-to-ASL Gloss translation algorithm needs additional modification that includes complex linguistic rules. The translation system needs better linguistic rules for ASL grammar because this unique structure makes generated sequences of sign language expression less natural and accurate.

Using authentic selected keyframes from video databases should replace synthetic data as a primary data source for creating vital improvements to the system. Real keyframes from actual world environments deliver better motion data representation which enhances the quality of generated interpolated results. An enhanced capability to handle different key point distance patterns within the model framework will expand its functional range to work with diverse signer communication forms. The performance of the system will greatly improve through the application of a high-quality motion-captured training dataset since this type of dataset delivers precise key point annotations which reduce detection errors in coordinates. The execution of these system updates will result in better accuracy and fluid motion and real-world effects for 3D ASL animated productions.

B. Conclusion

In this research, a feasible combination of NLP, Deep Learning, and 3D modeling was demonstrated to perform reliable real-time speech translation into ASL. By employing BERT, LSTM, and 3D pose estimation techniques, the developed system provided solutions for the complexities involved in ASL gestures, involving facial expressions and body language, overcoming the limitations posed by traditional 2D translation systems. The outcome is promising in terms of the accuracy noted and the access level reached, though there are still questions about computational cost and scalability. Further improvements must be concerned with providing better optimizations for making the system readily available and scalable in achieving its fullest potential in real-world applications and, with a continued focus on ethical issues and inclusivity, assured for the Dissing and Hard-of-Hearing (DHH) community.

References

[1] D. Bragg, O. Koller, M. Bellard, L. Berke, P. Boudreault, A. Braffort, N. Caselli, M. Huenerfauth, H. Kacorri, T. Verhoef, C. Vogler, and M. R. Morris, "Sign language recognition, generation, and translation: An interdisciplinary perspective," 2019.

[2] S. Ebling and J. Glauert, *Building a Swiss German Sign Language avatar with JASigning and evaluating it among the Deaf community*, 05 2015, vol. 15.

[3] D. Das Chakladar, P. Kumar, S. Mandal, P. Roy, M. Iwamura, and B.-G. Kim, "3d avatar approach for continuous sign movement using speech/text," *Applied Sciences*, vol. 11, p. 3439, 04 2021.

[4] B. Saunders, N. C. Camgoz, and R. Bowden, "Adversarial training for multi-channel sign language production," 2020. [Online]. Available: https://arxiv.org/abs/2008.12405

[5] D. Konstantinidis, K. Dimitropoulos, and P. Daras, "A deep learning approach for analyzing video and skeletal features in sign language recognition," 08 2018.

[6] S. Stoll, N. Camgoz, S. Hadfield, and R. Bowden, "Text2sign: Towards sign language production using neural machine translation and generative adversarial networks," *International Journal of Computer Vision*, 04 2020.

[7] R.-H. Liang and M. Ouhyoung, "A real-time continuous gesture recognition system for sign language," 05 1998, pp. 558 – 567.

[8] I. Infantino, R. Rizzo, and S. Gaglio, "A framework for sign language sentence recognition by commonsense context," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 5, pp. 1034–1039, 2007.

[9] J. Forster, C. Oberdörfer, O. Koller, and H. Ney, "Modality combination techniques for continuous sign language recognition," in *Iberian Conference on Pattern Recognition and Image Analysis*, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:11640638

[10] W. Gao, G. Fang, D. Zhao, and Y. Chen, "Transition movement models for large vocabulary continuous sign language recognition," in *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, 2004, pp. 553–558.

[11] G. Fang, W. Gao, and D. Zhao, "Large-vocabulary continuous sign language recognition based on transition-movement models," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 37, no. 1, pp. 1–9, 2007.

[12] O. Koller, H. Ney, and R. Bowden, "Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3793–3802.

[13] J. Pu, W. Zhou, and H. Li, "Iterative alignment network for continuous sign language recognition," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4160–4169.

[14] D. Konstantinidis, K. Dimitropoulos, and P. Daras, "A deep learning approach for analyzing video and skeletal features in sign language recognition," in *2018 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2018, pp. 1–6.

[15] D. Kothadiya, C. Bhatt, K. Sapariya, K. Patel, A.-B. Gil-González, and J. M. Corchado, "Deepsign: Sign language detection and recognition using deep learning," *Electronics*, vol. 11, no. 11, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/11/1780

[16] B. T. Lowerre, "The harpy speech recognition system," 1976. [Online]. Available: https://api.semanticscholar.org/CorpusID:61409851

[17] K. M. Ravikumar, R.Rajagopal, and H.C.Nagaraj, "An approach for objective assessment of stuttered speech using mfcc features," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:214736991

[18] K. M. Ravikumar, B. Reddy, R. Rajagopal, and H. C. Nagaraj, "Automatic detection of syllable repetition in read speech for objective assessment of stuttered disfluencies," *World Academy of Science, Engineering and Technology, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, vol. 2, pp. 2142–2145, 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:7740327

[19] D. Mitrović, M. Zeppelzauer, and H. Eidenberger, "On feature selection in environmental sound recognition," in *2009 International Symposium ELMAR*, 2009, pp. 201–204.

[20] C. Wang, Y. Wu, Y. Wu, Y. Qian, K. Kumatani, S. Liu, F. Wei, M. Zeng, and X. Huang, "Unispeech: Unified speech representation learning with labeled and unlabeled data," *ArXiv*, vol. abs/2101.07597, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:231639364

[21] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao, J. Wu, L. Zhou, S. Ren, Y. Qian, Y. Qian, M. Zeng, and F. Wei, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, pp. 1505–1518, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:239885872

[22] M. Liu, Y. Wang, J. Wang, J. Wang, and X. Xie, "Speech enhancement method based on lstm neural network for speech recognition," in *2018 14th IEEE International Conference on Signal Processing (ICSP)*, 2018, pp. 245–249.

[23] V. Passricha and R. K. Aggarwal, "A hybrid of deep cnn and bidirectional lstm for automatic speech recognition," *Journal of Intelligent Systems*, vol. 29, pp. 1261 – 1274, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:86766369

[24] W. Han, Z. Zhang, Y. Zhang, J. Yu, C.-C. Chiu, J. Qin, A. Gulati, R. Pang, and Y. Wu, "Contextnet: Improving convolutional neural networks for automatic speech recognition with global context," 2020.

[25] S. Papi, M. Gaido, M. Negri, and M. Turchi, "Speechformer: Reducing information loss in direct speech translation," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021. [Online]. Available: http://dx.doi.org/10.18653/v1/2021.emnlp-main.127

[26] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "data2vec: A general framework for self-supervised learning in speech, vision and language," 2022.

[27] C. Wang, S. Chen, Y. Wu, Z.-H. Zhang, L. Zhou, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li, L. He, S. Zhao, and F. Wei, "Neural codec language models are zero-shot text to speech synthesizers," *ArXiv*, vol. abs/2301.02111, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:255440307

[28] J. Gao, T. Shen, Z. Wang, W. Chen, K. Yin, D. Li, O. Litany, Z. Gojcic, and S. Fidler, "Get3d: A generative model of high quality 3d textured shapes learned from images," 2022.

[29] J. Xu, X. Wang, W. Cheng, Y.-P. Cao, Y. Shan, X. Qie, and S. Gao, "Dream3d: Zero-shot text-to-3d synthesis using 3d shape prior and text-to-image diffusion models," 2023.

[30] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," 2022.

[31] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, "Magic3d: High-resolution text-to-3d content creation," 2023.

[32] A. Raj, S. Kaza, B. Poole, M. Niemeyer, N. Ruiz, B. Mildenhall, S. Zada, K. Aberman, M. Rubinstein, J. Barron, Y. Li, and V. Jampani, "Dreambooth3d: Subject-driven text-to-3d generation," 2023.

[33] R. Chen, Y. Chen, N. Jiao, and K. Jia, "Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation," 2023.

[34] S. Mo, E. Xie, R. Chu, L. Yao, L. Hong, M. Nießner, and Z. Li, "Dit-3d: Exploring plain diffusion transformers for 3d shape generation," 2023.

[35] N. C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden, "Neural sign language translation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7784–7793.

[36] R. Costa, T. Araujo, M. Aschoff, V. Veríssimo, R. Andrade, S. Vieira, A. Santos, G. Souza Filho, M. Soares, and V. Hanael, "Towards an open platform for machine translation of spoken languages into sign languages," *Machine Translation*, vol. 33, 12 2019.

[37] S. Stoll, N. Camgoz, S. Hadfield, and R. Bowden, "Text2sign: Towards sign language production using neural machine translation and generative adversarial networks," *International Journal of Computer Vision*, 04 2020.

[38] B. Saunders, N. C. Camgoz, and R. Bowden, "Adversarial training for multi-channel sign language production," 2020.

[39] L. Chaudhary, T. Ananthanarayana, E. Hoq, and I. Nwogu, "Signnet ii: A transformer-based two-way sign language translation model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 12 896–12 907, 2023.

[40] S. Stoll, A. Mustafa, and J.-Y. Guillemaut, "There and back again: 3d sign language generation from text using back-translation," in *2022 International Conference on 3D Vision (3DV)*, 2022, pp. 187–196.

[41] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022. [Online]. Available: https://arxiv.org/abs/2212.04356

[42] M. Lewis, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[44] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013. [Online]. Available: https://arxiv.org/abs/1310.4546

[45] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," 2016. [Online]. Available: https://arxiv.org/abs/1607.01759

[46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[47] A. Alokla, W. Gad, W. Nazih, M. Aref, and A.-b. Salem, "Pseudocode generation from source code using the bart model," *Mathematics*, vol. 10, no. 21, 2022. [Online]. Available: https://www.mdpi.com/2227-7390/10/21/3967

[48] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2023. [Online]. Available: https://arxiv.org/abs/1910.10683

[49] J. Alammar, "The illustrated transformer," 2018, blog post. [Online]. Available: https://jalammar.github.io/illustrated-transformer/

[50] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. F. Aji, N. Bogoychev, A. F. T. Martins, and A. Birch, "Marian: Fast neural machine translation in c++," 2018. [Online]. Available: https://arxiv.org/abs/1804.00344

[51] A. S. Soliman, M. M. Hadhoud, and S. I. Shaheen, "Mariancg: a code generation transformer model inspired by machine translation," *Journal of Engineering and Applied Science*, vol. 69, no. 1, p. 104, 2022.

[52] S. Kreiss, L. Bertoni, and A. Alahi, "Openpifpaf: Composite fields for semantic keypoint detection and spatio-temporal association," 2021. [Online]. Available: https://arxiv.org/abs/2103.02440

[53] S. Lutz, R. Blythman, K. Ghosal, M. Moynihan, C. Simms, and A. Smolic, "Jointformer: Single-frame lifting transformer with error

prediction and refinement for 3d human pose estimation," 2022. [Online]. Available: https://arxiv.org/abs/2208.03704

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[55] T. Jiang, P. Lu, L. Zhang, N. Ma, R. Han, C. Lyu, Y. Li, and K. Chen, "Rtmpose: Real-time multi-person pose estimation based on mmpose," 2023. [Online]. Available: https://arxiv.org/abs/2303.07399

[56] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for building perception pipelines," 2019. [Online]. Available: https://arxiv.org/abs/1906.08172

[57] A. Duarte, S. Palaskar, L. Ventura, D. Ghadiyaram, K. DeHaan, F. Metze, J. Torres, and X. G. i Nieto, "How2sign: A large-scale multimodal dataset for continuous american sign language," 2021. [Online]. Available: https://arxiv.org/abs/2008.08143

[58] J. Schmidhuber, S. Hochreiter *et al.*, "Long short-term memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997.

[59] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4346–4354.

[60] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," 2015. [Online]. Available: https://arxiv.org/abs/1506.06724

[61] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[62] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. USA: Association for Computational Linguistics, 2002, p. 311–318. [Online]. Available: https://doi.org/10.3115/1073083.1073135

[63] F. Noorbehbahani and A. A. Kardan, "The automatic assessment of free text answers using a modified bleu algorithm," *Computers & Education*, vol. 56, no. 2, pp. 337–345, 2011.