American Sign Language Detection Using Deep Learning

Team 9
Project Option 1

Kaushika Uppu
014756859
kaushika.uppu@sjsu.edu

Nivedita Nair
018184777
nivedita.nair@sjsu.edu

Bhavya Jain
018208671
bhavya.jain@sjsu.edu

Abstract*—American Sign Language (ASL) is a rich, complex language, but it is understood by very few non-signers. This unfortunately leads to accessibility challenges, especially since human interpreters are not always available. Digital, automated interpreters can help bridge this gap. In this project, we aim to move beyond ASL image letter detection to interpret hand-gesture videos of ASL words using deep learning. We found that of all the models, the state-of-the-art UniSign outperformed the others with top-1, top-5, and top-10 accuracies of 40.5%, 73.00%, and 83.50%. The enhanced UniSign that we simplified the architecture of to reduce overfitting did not significantly improve accuracy, but did not significantly decline in performance either.

Keywords*—American Signal Language (ASL), deep learning, LSTM, MediaPipe, UniSign, VLM.

I. Introduction

American Sign Language, or ASL, serves as the primary mode of communication for millions of people who are deaf or hard of hearing. However, most non-signers do not know or understand the language, making it difficult for ASL users to communicate at times. These accessibility issues are deepened by the fact that human interpreters are not always available for translation, exacerbating communication problems even further. One way in which this issue can be addressed is through digital interpreters, namely automated pipelines that can serve as ASL translators.

The automatic recognition of ASL is thus a critical area of research with the potential to significantly enhance accessibility, improve human-computer interaction, and facilitate communication between the deaf and hearing communities. However, building robust and accurate ASL recognition systems remains a considerable technical challenge due to the temporal nature of signs, the subtle visual variations between similar signs, and the lack of large, consistently captured datasets.

In this project, we intend to integrate deep learning models into a pipeline that goes beyond image ASL letter detection and to interpreting words using hand-gesture videos. We aim to evaluate and analyze performance of multiple deep learning frameworks, including an initial baseline model of a Long Short-Term Memory (LSTM) network and moving towards state-of-the-art models such as UniSign and CTR-GCN.

II. Related Work

American Sign Language (ASL) recognition has been extensively studied using video-based deep learning, pose estimation, and sequence modeling methods. This project builds upon several key datasets, hand-tracking frameworks, and deep learning architectures that form the foundation for modern sign language understanding systems.

2.1 Datasets

The project primarily uses the Word-Level American Sign Language (WLASL) dataset, introduced in [1], which is one of the largest publicly available ASL video datasets. WLASL provides multiple vocabulary subsets and includes signer metadata, bounding-box annotations, and diverse signing styles. For computational efficiency, this project uses the WLASL100 subset. Additionally, the ASL Alphabet dataset from Kaggle is used to develop baseline models for static handshape recognition.

2.2 Hand Landmark Detection

Hand landmark extraction is performed using Google's MediaPipe Hands framework [2], which provides real-time detection of 21 three-dimensional keypoints per hand. MediaPipe serves as the backbone for preprocessing video samples by converting raw frames into structured sequences of hand landmarks (typically 30 frames × 21 landmarks × 3 coordinates). These pose-based representations enable efficient temporal modeling while significantly reducing computational overhead compared to full-frame video processing.

2.3 Deep Learning Architectures

2.3.1 I3D (Inflated 3D ConvNet)

Carreira and Zisserman [3] introduced I3D as a 3D convolutional extension of 2D ConvNets, pre-trained on the Kinetics dataset. I3D has become a standard baseline for video action recognition and is included in the WLASL benchmark for comparative analysis. While this project does not directly implement I3D, its presence as a baseline informs model evaluation and design choices.

2.3.2 Temporal Graph Convolutional Networks (TGCN)

Spatial-Temporal Graph Convolutional Networks (ST-GCN) [4] provide a graph-based approach to action recognition using human skeletal data. WLASL includes TGCN as a pose-based baseline. The principles of graph-based temporal modeling such as representing joints as nodes and their relationships as edges, influence the design of landmark-based recognition strategies explored in this project.

2.3.3 Transformer Architecture

The transformer architecture introduced by Vaswani et al. [5] revolutionized sequence modeling by replacing recurrent networks with multi-head self-attention. Its ability to capture long-range dependencies makes it particularly effective for modeling temporal hand landmark sequences.

This project implements a transformer-based model referred to as UniSign [6], which applies spatial-temporal multi-head attention across landmark frames. Position encoding, residual connections, and layer normalization are incorporated to achieve robust sequence classification.

2.3.4 Long Short-Term Memory Networks (LSTM)

LSTM networks [7] represent a classical approach to temporal data modeling. In this work, LSTMs serve as a baseline method for sequence classification of landmark trajectories, enabling comparison against more modern transformer-based architectures.

2.3.5 Transfer Learning Models

To evaluate static image recognition performance, MobileNetV3 [8] and EfficientNet [9] are employed as lightweight, high-performance CNN architectures. Both architectures are pre-trained on ImageNet and fine-tuned on ASL alphabet images to establish baseline accuracy for static hand gesture recognition.

2.4 Related Research

2.4.1 Sign Language Recognition

Recent work by Li et al. [10] explores cross-domain transfer learning, temporal semantic pyramid networks (TSPNet), and neural editing programs for sign language translation. This study highlights the challenges in modeling complex spatiotemporal patterns inherent in sign language and informs the architectural choices in this project.

2.4.2 Action Recognition and Video Understanding

Earlier contributions such as Two-Stream Networks [11] and 3D CNNs [12] established foundational approaches in video understanding. These techniques underpin modern architectures and motivate explorations into spatiotemporal deep learning.

2.4.3 Attention Mechanisms

Works such as Bahdanau et al. [13] and Devlin et al. [14] demonstrate the effectiveness of attention mechanisms in translation and representation learning. Their influence extends to sign language applications, providing theoretical backing for transformer-based models.

This project builds on the above foundation and introduces several key contributions:

1. Landmark-Based Sequence Modeling
   Conversion of video frames into landmark sequences reduces computational load and enables real-time recognition while preserving motion dynamics.

2. UniSign Transformer Architecture
   A custom spatial-temporal transformer architecture tailored for ASL recognition, leveraging attention mechanisms to model complex gesture dynamics.

3. Enhanced Feature Engineering
   The project incorporates not only raw landmark coordinates but also derived features such as velocity, acceleration, inter-landmark distances, and angular relationships. Per-hand normalization improves cross-signer robustness.

4. Comprehensive Evaluation Framework
   A number of model architectures—LSTM, transformer-based, and transfer learning—are compared against state-of-the-art models such as CTR-GCN. Top-K accuracy metrics are used to better assess practical usability.

III. Implementation and Model Designs

The dataset that we chose for the video detection was the Word-Level American Sign Language (WLASL) dataset [1], but more specifically, the WLASL-100 subset of this dataset. This consists of approximately 2,038 videos of the top-100 most frequent ASL words. This dataset already contains test, train, and validation splits.

In terms of preprocessing, we first created a label map that mapped words to labels. Next, for each video in the dataset, landmarks were extracted using the MediaPipe Holistic pipeline. This resulted in 21 landmarks for each hand and 33 body pose landmarks. Body pose landmarks were extracted because for ASL words, simply looking at hand position is not always enough, as some words depend on the position relative to the body. Then, data was cleaned and missing hand frames were interpolated, and next, normalized. Finally, features were flattened into a 225-dimensional array—75 landmarks x 3 coordinates per landmark.

For the baseline model, we chose to implement a Bidirectional LSTM (BiLSTM) network with three layers. A summary of the model is pictured below (Fig. 1). We trained this model for 80 epochs using the Adam optimizer with a learning rate of 2e-4, a weight decay of 1e-4, a gradient clipping value of 0.5, and a dropout value of 0.4.

```
LSTM(
  (lstm): LSTM(225, 256, num_layers=3, batch_first=True, dropout=0.4, bidirectional=True)
  (fc): Linear(in_features=512, out_features=100, bias=True)
)
```

Fig. 1. Diagram of BiLSTM baseline network architecture.

One of the state-of-the-art models that we implemented was CTR-GCN, which is specifically designed for

skeleton-based action recognition. It models relationships between body joints and hand landmarks as a graph and captures temporal dynamics across frames, but dynamically learns graph structure instead of using a fixed one. We defined the CTR-GCN model in our project based on the original model on Github [15]. A diagram of this model is shown below, taken from the paper for CTR-GCN (Fig. 2). The model was trained for up to 80 epochs using Adam optimizer with a learning rate of 1e-3, weight decay of 1e-4, batch size of 32, and a gradient clipping value of 5.0.

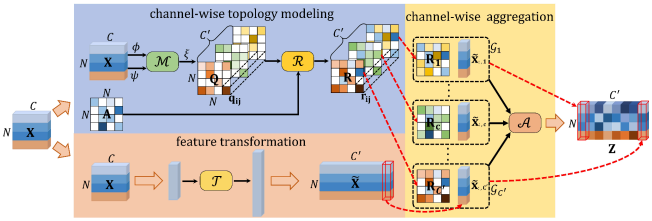

Fig. 2. Model architecture for CTR-GCN.

Next, we implemented another state-of-the-art framework, UniSign [6]. This is a transformer-based encoder architecture for spatial-temporal sequence classification. The model consists of an input projection layer that maps 225-dimensional features to a 256-dimensional embedding space (d_model), followed by learnable positional encodings for the 32 temporal frames. The diagram below, taken from the UniSign paper, showcases the complete original model architecture (Fig. 3).

The core architecture of UniSign comprises 4 transformer encoder blocks, each containing multi-head self-attention with 8 heads, a feed-forward network with 512 units, GELU activation, residual connections, and layer normalization. The temporal sequence is aggregated using global average pooling, followed by a two-layer classification head with batch normalization and dropout regularization, outputting a probability distribution over 100 ASL word classes. The model was trained for up to 80 epochs using AdamW optimizer with a learning rate of 2e-4, weight decay of 1e-4, and a batch size of 32, with early stopping (patience=15) and checkpointing.

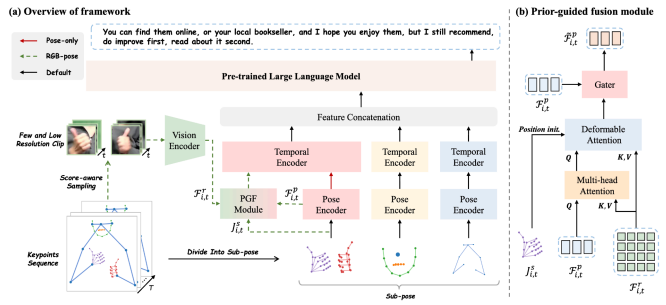

Fig. 3. Network structure for UniSign.

Finally, the Vision-Language Model (VLM) approach leverages the CLIP ViT-B/32 visual encoder to extract pretrained image embeddings directly from raw RGB video frames, offering a fundamentally different strategy from landmark-based models such as UniSign and CTR-GCN. Each WLASL-100 video was converted into a sequence of 32 uniformly sampled frames, augmented with random cropping, rotation, and brightness adjustments to improve generalization. The CLIP encoder produced 768-dimensional embeddings per frame, which were passed through a lightweight transformer-based classification head trained on top of the frozen visual features.

We implemented an inference application using Streamlit, enabling end-to-end processing of user-uploaded videos. The system samples video frames, extracts embeddings using the CLIP backbone, and feeds the sequence into our regularized classifier to display predictions.

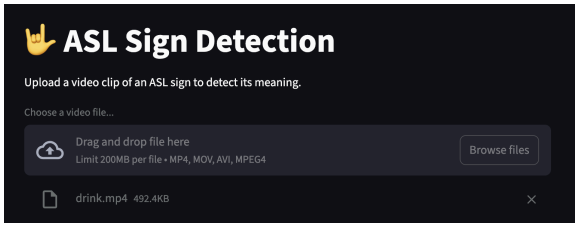

Fig. 4. Interactive ASL detection user interface created using Streamlit.

IV. Task Distribution and Contributions

Task	Contributor(s)
Feature Engineering - letter ASL	Kaushika Uppu
Training - ASL	Nivedita Nair; Kaushika Uppu
Feature Engineering WLASL	Kaushika Uppu; Bhavya Jain
Training WLASL - Unisign	Nivedita Nair
Training WLASL - CTR-GCN	Kaushika Uppu
Training WLASL - Unisign feature enhancement	Bhavya Jain
WLASL - VLM	Bhavya Jain
Documentation + Presentation	Kaushika Uppu; Bhavya Jain; Nivedita Nair

V. Evaluation and Results

5.1 Evaluation Metrics

The system was evaluated using a combination of classification metrics, training metrics, and computational performance metrics.

5.1.1 Classification Metrics

To assess recognition performance across the 100 ASL classes, the following metrics were computed:

1. Top-1 Accuracy – the standard accuracy measuring exact class match
2. Top-5 Accuracy – proportion of samples where the correct label appears among the top five predicted classes
3. Top-10 Accuracy – proportion of samples where the correct label appears among the top ten predicted classes

Top-K metrics are essential for sign language applications since near-correct predictions are also useful in real-world assistive systems. A confusion matrix was also generated to visualize misclassifications and identify classes frequently confused with each other.

5.1.2 Training Metrics

During training, the following metrics were monitored:

1. Training Loss (Sparse Categorical Crossentropy)
2. Validation Loss
3. Learning Rate (varied using cosine annealing schedule)

These metrics help analyze convergence, detect overfitting, and evaluate optimization behavior.

5.2 Model Performance Results

5.2.1 LSTM (Baseline)

Metric	Score
Top-1 Accuracy	22.50%
Top-5 Accuracy	57.50%
Top-10 Accuracy	74.00%

As seen in the table, the accuracies are not very high for LSTM. However, this was our baseline model, and we expected the state-of-the-art models to perform better.

5.2.2 UniSign Transformer

Metric	Score
Top-1 Accuracy	40.50%
Top-5 Accuracy	73.00%
Top-10 Accuracy	83.50%
Test Loss	2.5297

Architecture Summary:

- 4 transformer layers
- 256-dimensional embeddings
- 8 attention heads
- Sequence length: 32 frames
- Input features: 675 (landmarks + velocity + acceleration)

We decided to simplify the architecture of the unisign transformer to reduce the impact of overfitting. The metrics for this simplified architecture are as follows:

Metric	Score
Top-1 Accuracy	39.00%
Top-5 Accuracy	73.50%
Top-10 Accuracy	81.50%
Test Loss	2.7337

Architecture Summary:

- 2 transformer layers
- 128-dimensional embeddings
- 4 attention heads
- Sequence length: 32 frames
- Input features: 225 (landmarks + velocity + acceleration)

The simplification of the model did not help improve the model. Only Top-5 accuracy is comparable while other metrics are lesser for the simplified model.

As we can see through both the architectures, the transformer significantly outperforms LSTM due to better ability to model long-range temporal dependencies.

5.2.3 CTR-GCN

Metric	Score
Top-1 Accuracy	36.00%
Top-5 Accuracy	67.00%
Top-10 Accuracy	77.00%

As seen above, while the CTR-GCN state-of-the-art model outperforms the LSTM baseline model, it does not reach the performance of the UniSign transformer-based model. Since the CTR-GCN model utilizes graphs to model skeletal-based actions, it is likely able to capture nuances in ASL words better than LSTM can. However, the transformer is still likely able to model longer-range dynamics better.

5.2.4 VLM - CLIP -ViT batch 32

The results for the Vision-Language Model (VLM) approach are shown below. While it does perform better than baseline, it is not by a large margin. Therefore, it does not outperform either CTR-GCN or UniSign.

Metric	Score
Top-1 Accuracy	25.19%
Top-5 Accuracy	60.47%
Top-10 Accuracy	74.03%

Architecture Summary:

- CLIP ViT-B/32 pretrained visual encoder
- Input: 32 RGB frames per video
- Frame embedding size: 768
- Temporal classifier: 2-layer transformer with dropout
- Sequence length: 32 frames
- Augmentations: random crop, rotation, brightness jitter

5.2.5 Comparative Performance

Model	Top-1	Top-5	Top-10
LSTM	22.50%	57.50%	74.00%
CTR-GCN	36.00%	67.00%	77.00%
UniSign Transformer	40.50%	73.00%	83.50%
UniSign Enhanced	39.00%	73.50%	81.50%
VLM	25.19%	60.47%	74.03%

5.3 Visualizations

5.3.1 Loss and Accuracy Curves

The figures below reveal the loss and accuracy curves for all of the models that we implemented.

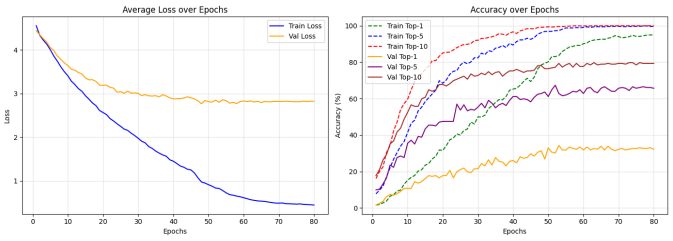

Fig. 5. Training vs validation loss and accuracy of the LSTM baseline model.

As seen above on Fig. 5 on the right, the accuracies of the LSTM module are not very high, especially for Top-1 accuracy.

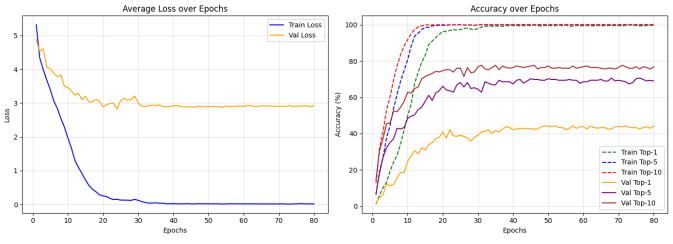

Fig. 6. Training vs validation loss and accuracy of the CTR-GCN model.

Fig. 6 shows the loss and accuracy curves that we got for the CTR-GCN framework. Compared to the LSTM, the CTR-GCN model gets higher accuracies and reaches 100% accuracy during training, though it does not quite get there during validation.

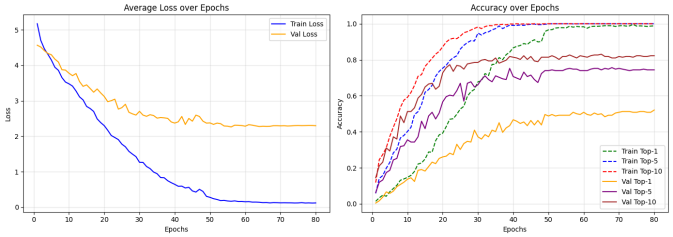

Fig. 7. Training vs validation loss and accuracy of the Unisign Transformer.

As seen in the plot above (Fig. 7), the UniSign transformer model outperformed the other two models significantly. Although the CTR-GCN model seemed to learn during training slightly faster, UniSign gets a higher overall validation accuracy.

5.3.2 Confusion Matrices

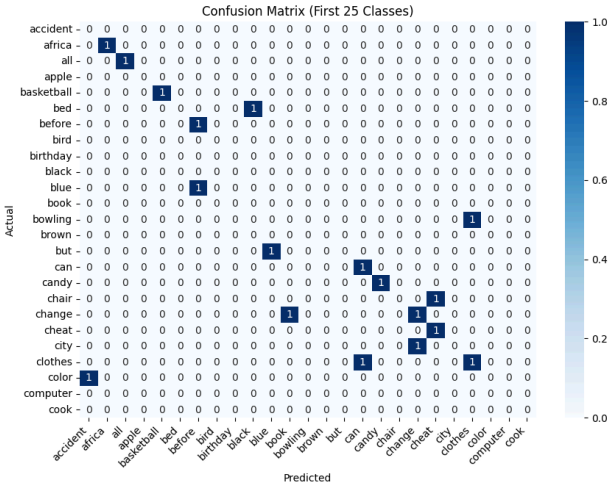

Fig. 8. Confusion matrix showing class-wise performance for WLASL-100 of the LSTM model.

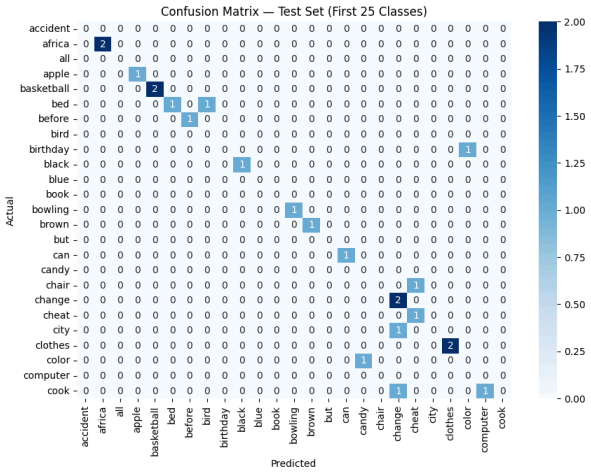

Fig. 9. Confusion matrix showing class-wise performance for WLASL-100 of the CTR-GCN model.

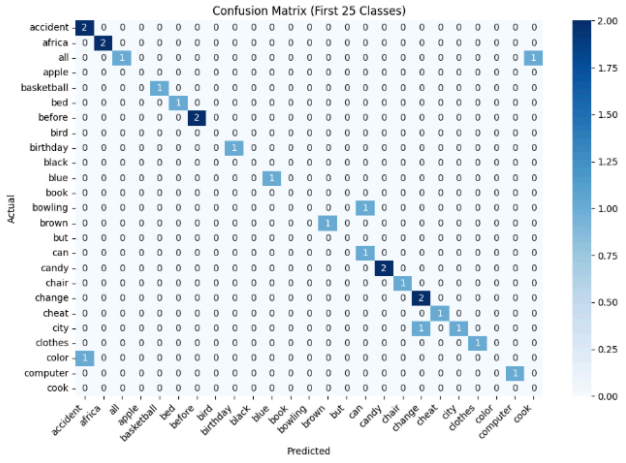

Fig. 10. Confusion matrix showing class-wise performance for WLASL-100 from UniSign model.

5.3.3 Classification Report

Classification Report:

	precision	recall	f1-score	support
accident	0.40	0.67	0.50	3
africa	0.67	1.00	0.80	2
all	0.33	0.50	0.40	2
apple	0.00	0.00	0.00	1
basketball	0.33	0.50	0.40	2
bed	1.00	0.50	0.67	2
before	0.50	0.67	0.57	3
bird	0.00	0.00	0.00	2
birthday	0.33	0.50	0.40	2
black	0.00	0.00	0.00	3
blue	0.50	0.50	0.50	2
book	0.00	0.00	0.00	0
bowling	0.00	0.00	0.00	2
brown	1.00	0.50	0.67	2
but	0.00	0.00	0.00	2
can	0.50	1.00	0.67	1
candy	0.33	0.67	0.44	3
chair	1.00	0.50	0.67	2
change	0.50	1.00	0.67	2
cheat	0.50	1.00	0.67	1

Fig. 11. Classification report showing precision, recall, F1-score and support of the Unisign model.

5.4 Reproducibility

The libraries required to run the notebooks were installed using pip commands. The following libraries were installed: tensorflow, mediapipe, scikit-learn, numpy, and opencv-python. Once these libraries are installed, the more detailed commands and implementations can be found at our Github repository: ASL Detection Using Deep Learning.

References

[1] D. Li, C. Rodriguez, X. Yu, and H. Li, "Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, 2020, pp. 1459–1469.

[2] "Hand landmarks detection guide for Python," *Google AI for Developers*, 2025. https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker/python#live-stream.

[3] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," *arXiv.org*, 2017. https://arxiv.org/abs/1705.07750.

[4] S. Yan, Y. Xiong, and D. Lin, "Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition," *arXiv:1801.07455 [cs]*, Jan. 2018, Available: https://arxiv.org/abs/1801.07455.

[5] A. Vaswani *et al.*, "Attention Is All You Need," *Cornell University*, Jun. 12, 2017. https://arxiv.org/abs/1706.03762

[6] Z. Li, W. Zhou, W. Zhao, K. Wu, H. Hu, and H. Li, "Uni-Sign: Toward Unified Sign Language Understanding at Scale," *arXiv* preprint arXiv:2501.15187, 2025. [Online]. Available: https://arxiv.org/pdf/2501.15187.

[7] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: https://doi.org/10.1162/neco.1997.9.8.1735.

[8] A. Howard *et al.*, "Searching for MobileNetV3," *arXiv.org*, 2019. https://arxiv.org/abs/1905.02244

[9] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *arXiv.org*, May 28, 2019. https://arxiv.org/abs/1905.11946

[10] D. Li *et al.*, "TSPNet: Hierarchical Feature Learning via Temporal Semantic Pyramid for Sign Language Translation," *arXiv.org*, 2020. https://arxiv.org/abs/2010.05468.

[11] K. Simonyan and A. Zisserman, "Two-Stream Convolutional Networks for Action Recognition in Videos," *arXiv.org*, 2014. https://arxiv.org/abs/1406.2199

[12] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," *arXiv.org*, 2014. https://arxiv.org/abs/1412.0767

[13] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *arXiv.org*, Sep. 01, 2014. https://arxiv.org/abs/1409.0473

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *ArXiv*, Oct. 11, 2018. https://arxiv.org/abs/1810.04805

[15] Uason-Chen, "CTR-GCN: Channel-wise Topology Refinement Graph Convolution for Skeleton-Based Action Recognition," *GitHub* repository. [Online]. Available: https://github.com/Uason-Chen/CTR-GCN.