National College of Ireland

Word-Based American Sign Language Recognition Using 2D CNN with Temporal Shift Module and Attention

MSc Research Project
MSc in Data Analytics

Semanta Das
Student ID: X23338806

School of Computing
National College of Ireland

Supervisor: Prof. Vikas Tomer

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Semanta Das
Student ID:	X23338806
Programme:	MSc in Data Analytics
Year:	2024-2025
Module:	MSc Research Project
Supervisor:	Prof. Vikas Tomer
Submission Due Date:	11/08/2025
Project Title:	Word-Based American Sign Language Recognition Using 2D CNN with Temporal Shift Module and Attention
Word Count:	4500
Page Count:	23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Semanta Das
Date:	14th September 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Word-Based American Sign Language Recognition Using 2D CNN with Temporal Shift Module and Attention

Semanta Das
X23338806

Abstract

This work presents a lightweight American Sign Language (ASL) word recognition system built on a 2D CNN backbone (MobileNet-V2) enhanced with Temporal Attention and a Temporal Shift Module (TSM). The system was trained and evaluated on the 100-word subset of the MS-ASL dataset. A four-stage incremental development process was followed, where the weights from each stage were transferred to initialize the next which enables progressive learning. The process began with a frame-level pretraining in Stage A. From Stage B, temporal attention was applied, which helped to get better accuracy. In Stage C, stronger data augmentation was applied, with a dropout rate of 0.5 which helped reduce overfitting. Finally, in the Stage D, TSM was introduced with attention which enhanced temporal modelling. The final model achieved a top-1 accuracy of 51.97% on the MS-ASL100 dataset. Compared with model like 3D CNNs and Transformer, my proposed model is not that much effective, as we are only working with 2D kernel.

1 Introduction

Sign Language is a fully developed visual language that allows deaf individuals to communicate through body movement such as hand gestures, facial expressions, or using other body parts. The deaf community of United States and several parts of Canada use American Sign Language (ASL) as their main language. It has its own grammar, which makes it different from spoken English. ASL is one of the most widely used sign language all over the word and one of the most studied language.

To mitigate the gap between Deaf and hearing people, researchers have been working on Sign Language Recognition (SLR) systems. These systems' main goal to translate sign language into text or speech so that communication can be occurred more quickly and effectively. Public datasets such as MS-ASL (Microsoft American Sign Language) and WLASL (Word-Level American Sign Language) are contributing to the field of SLR, By using these word-based video dataset researchers have been doing a lot of work, which comes in favor in SLR system.

Many studies have been conducted on static ASL such as alphabets or numbers. However, these are not effective for real-time communication. For this reason, word-level sign language recognition using video datasets has emerged in the field of SLR. MS-ASL dataset was first published in 2018 with a research paper, where an Inflated 3D (I3D) CNN was used and achieved good score for MS-ASL 100 class (for 100 words), which

was 81.76% top-1 accuracy and for MS-ASL1000 (full dataset) accuracy was 57.7% by learning spatio-temporal features directly from the videos (Vaezi Joze and Koller; 2019). More recently, transformer-based systems that fuse RGB, pose and even text cues have pushed performance further (such as Uni-SIGN: 93.8% on MS-ASL100 and 78.2% on MS-ASL1000) (Li et al.; 2025) and large multimodal pre-training (SL-1.5M) reports 74.1% on MS-ASL1000 (Zhou et al.; 2024).

However, these high-performing model come with a cost such as 3D CNNs require heavy computation and memory while Transformers with pose and text fusion and large pre-training need powerful GPUs and long training times. On the other hand, Lightweight 2D CNN-based approaches do exist such as, MobileNet pipelines or CNN with attention, but most of systems are tested in limited conditions. For example, A model including MobileNetV2 with an LSTM and temporal attention was used to train WLASL100 dataset and achieved 84.65% accuracy (Kumari and Anand; 2024). Also a multi-channel CNN trained on a small Kinect-based alphabet dataset of static hand shapes, but this approach does not include any videos data (Mudumbai et al.; 2024). Adding RNNs like LSTM to CNN models increases complexity, making them heavier than CNN only approaches with TSM and attention.

Research gap. Despite having lots of progress there is no systematic study showing whether a lightweight 2-D backbone with simple temporal mechanisms such as attention and Temporal Shift Module can approach the accuracy of heavyweight models like 3D CNNs and Transformers using MS-ASL dataset. Our focus is building a 2D lightweight model using temporal mechanisms such as TSM and attention and check if model can produce comparable results like 3D CNNs or Transformers.

Research question. *Can a lightweight 2D CNN with Temporal Shift Modules (TSM) and attention, achieve sign-language recognition performance comparable to heavyweight models like 3D CNNs and transformers?*

Our contributions. To answer this question, we propose and study which is based on a MobileNetV2 backbone with temporal modeling:

- **Landmark-aware cropping** To reduce background landmark awareness, cropping techniques can be implemented by using MediaPipe. By doing this, model can focus more on the necessary areas such as hands and face, inspired by (Palaniappan et al.; 2025; Li et al.; 2025).

- **Progressive temporal modeling** Adopt progressive temporal modeling, starting with frame-level pretraining, then adding temporal attention and finally adding a Temporal Shift Module (TSM) for efficient temporal reasoning.

- **Regularization for low-shot classes** To handle low short classes strong data regularization and class imbalance sampling can be applied, which can also help to mitigate overfitting (Kumari and Anand; 2024).

Paper structure. Here is the short outline is given of our paper:

Title and Abstract → Introduction 5 (background, motivation, gap, question, contributions) → Related Work 1 (3D CNNs, 2D CNNs, Transformers, hybrid CNN-RNN, landmark methods, summary table) → Methodology 3 (data collection and preparation, modelling Stages A–D, evaluation) → Design Specification 4 (input preparation, architectures) → Implementation 5(tools, hardware, training, optimizer, test pipeline) → Evaluation 6 (stage-wise results, comparison, discussion) → Conclusion and Future Work 7 → References.

2 Related Work

In this section, I review important key parts of sign language recognition research, specially focused more on deep learning methods. Several researched was performed on various model architectures, training strategies using videos and static image-based sign data. Understanding these existing methods is essential to make some contributions in this field by proposing our model system.

2.1 Convolutional Backbones: Heavy 3D CNNs and Lightweight 2D CNNs

Early word-level Sign Language Recognition (SLR) system used Inflated 3D CNN (I3D) and achieved strong score on Microsoft American Sign Language (MS-ASL, a word-based dataset), where top-1 score for MS-ASL100 (100 classes or video-based words) is 81.8% and 57.7% for MS-ASL1000 (full dataset). This research performed by (Vaezi Joze and Koller; 2019). (Li et al.; 2020) also conducted research and achieved 65.9% on WLASL(Worl-Level American Sign Language Dataset) 100 class and 32.5% on WLASL2000(full dataset) class. Later on SEVNet (3D CNN, depth-wise and factorised) reached to 75% top-1 accuracy on Kinetics400 dataset (Wang and Yang; 2025). Being lighter than the I3D CNN it achieved really good score. But The Temporal Shift Module (TSM) (Lin et al.; 2019) later showed that shifting a small fraction of feature channels across neighbouring frames can replace explicit 3D convolutions. It allows 2D backbones to learn temporal relations with less complexity and architecture. MobileNet-V2 (a lightweight 2D CNN) have been explored for frame-level sign recognition. (Mudumbai et al.; 2024) reached perfect accuracy on a small Kinect-based letter corpus, but ignored motion and did not go beyond static alphabets. Our work keeps the computational frugality of MobileNet-V2 yet augments it with temporal attention (Stage B–D) and a TSM layer (Stage D) using the TSM technique like (Lin et al.; 2019) them, effectively narrow down the gap between heavy 3D models and 2D models.

2.2 Transformer, multimodal and large-scale pre-training approaches

Transformer is now dominating in the field of SLR. For example, A transformer called Uni-SIGN, it combines RGB images, pose heatmaps, and simplified gloss labels into a single transformer model and produces high accuracy of 93.8% on MS-ASL100 and 78.2% on MS-ASL1000 (Li et al.; 2025). Natural-Language-Assisted Sign Language Recognition(NLA-SLR) is another approach that uses language hints by breaking glosses (written representation of signs) into words and passing them into a transformer to improve performance (Zuo et al.; 2023). Additionally, SL-1.5M (which stands for Sign-Language 1.5 Million, a huge pre-training dataset) first trains a transformer with masked poses and text-sign contrastive learning, then fine-tunes it, and received 74.1% accuracy on MS-ASL1000 and 80.2% on the Non-Manual Features Chinese Sign Language(NMFs-CSL) dataset (Zhou et al.; 2024). Though these advanced models perform really well, these models require heavy computational resources as these are heavy model. On the other hand, the method used for this research is a much simpler 2D CNN MobileNetV2 model combined with Attention and Temporal Shift Module (TSM) technique. Even with fewer resources, it can provide good performance on 100 words (classes) for American Sign

Language by using some technique like landmark-based cropping, balanced-batches and strong data augmentation.

2.3 Hybrid CNN-RNN and Attention Frameworks

To handle longer sequences of motion, researchers often combine convolutional neural networks (CNN) with recurrent neural networks (RNN) such as Long Short-Term Memory(LSTM). For example, 2 layers of Bi-directional LSTMs was used by (Chophuk et al.; 2022) to analyze Leap-Motion skeleton features. Tadele et al.; (2025) combained a basic CNN with LSTM for Ethiopian Sign Language recognition (Tadele et al.; 2025). Attention methods with RNNs can provide good outcomes, such as Kumari et al.; (2024) combined MobileNetV2 with a Bi-LSTM and added attention to focus on important features and achieved 84.65% accuracy for WLASL100(Kumari and Anand; 2024) . Our Stage B model builds on almost similar idea, enhancing it with landmark-cropped images and regularization techniques. Maruyama et al.; (2024) took attention further by combining RGB and optical-flow using I3D backbone and skeleton data (Maruyama et al.; 2024), but their approach is heavier in computational cost and also complex than ours.

2.4 Landmark-guided and Region-focused Methods

In sign language, features such as hand shapes and facial expressions can provide critical information. For this reason, different kind of methods are used that focus on cropping images around detected landmarks so that irrelevant background can be reduced. For example, Palaniappan et al. (2025) used MediaPipe to detect landmarks and cropped hand and face areas, combining them with ResNet-LSTM (Palaniappan et al.; 2025). Similarly, UniSIGN guides its model with pixel-level landmark information (Li et al.; 2025). MediaPipe is also used in our method by prioritizing cropping around the hands and providing additional padding to retain important context. This simple landmark-based cropping method, combined with strong data augmentation, achieves significant accuracy improvements without heavy computation.

2.5 Summarize Table (Table: 1)

In this table a summary of the models and datasets used in this sign language recognition field is given. Though we have already described models and datasets used above, it can make the overview more suitable.

3 Methodology

This section describes the research methodology that is followed in this project. This methodology's steps followed detailing from the data preparation to model evaluation. Deep learning model is used in this project with four stage where each stage build using the outcomes of the previous stage.(Figure 1).

3.1 Data Collection

The Microsoft American Sign Language (MS-ASL) dataset, which originally contains 1000 classes, was downloaded from the official Microsoft Website and also published a

Table 1: Summary of existing models and datasets used in sign language recognition studies

Authors & Year	**Best-Performing Model**	**Dataset & Test Subset**	**Accuracy**
(Prachi Rawat et al.; 2025)	2-layer LSTM(SELU)	Custom ISL (11 gestures, 60 videos)	96.97%
(Palaniappan et al.; 2025)	ResNet50 + LSTM	Sign-Language MNIST (24 letters)	98.01%
(Wang and Yang; 2025)	SEVNet	SSV1/2, Kinetics-400, MMIT, FineGym	75%(K400), 62.4%(SSV2)
(Li et al.; 2020)	I3D	WLASL	65.89%(WLASL100) and 32.48%(WLASL2000)
(Chophuk et al.; 2022)	2-layer BiLSTM	Leap-Motion corpus	97.34% and 96.99%
(Kumari and Anand; 2024)	MobileNetV2 + attention-LSTM	WLASL100	84.65%
(Izutov; 2020)	S3D-MobileNetV3 + ST-attention	MS-ASL	85.00% (MS-ASL100) and 45.65% (MS-ASL1000)
(Lin et al.; 2019)	TSM + ResNet-50	SSV1/2, Kinetics-400	: 64.3% (SSV2), 74.1%(K400)
(Zuo et al.; 2023)	VKNet + Lang-aware Language-Strem	MS-ASL, WLASL, NMFs-CSL	90.49% (MS-ASL100), 72.56% (MS-ASL1000), 91.47% (WLASL100), 61.05% (WLASL2000), 83.4% (NMFs-CSL).
(Zhou et al.; 2024)	Multimodal	SL-1.5M → MS-ASL1000, NMFs-CSL, SLR500	74.07% (MS-ASL1000), 80.2% (NMF-CSL), 97.7% (SLR500)
(Mudumbai et al.; 2024)	RGB+Depth+Skeleton CNN	Kinect in-house	100%
(Babu et al.; 2024)	MobileNetV1 + SVM	ASL static (36 classes)	99.5%
(Li et al.; 2025)	Uni-Sign (RGB + pose + gloss)	MS-ASL, WLASL, CSL-Daily	93.79% (MS-ASL100)
(Vaezi Joze and Koller; 2019)	I3D	MS-ASL 100–1000	81.76% (MS-ASL100) and 57.69% (MS-ASL1000)
Bekalu Tadele et al. (2025) (Tadele et al.; 2025)	Hybrid CNN + LSTM	Ethiopian SL (10 words)	95%
Mizuki Maruyama et al. (2024) (Maruyama et al.; 2024)	Multi-stream MSNN	WLASL, MS-ASL	83.9% (MS-ASL100)

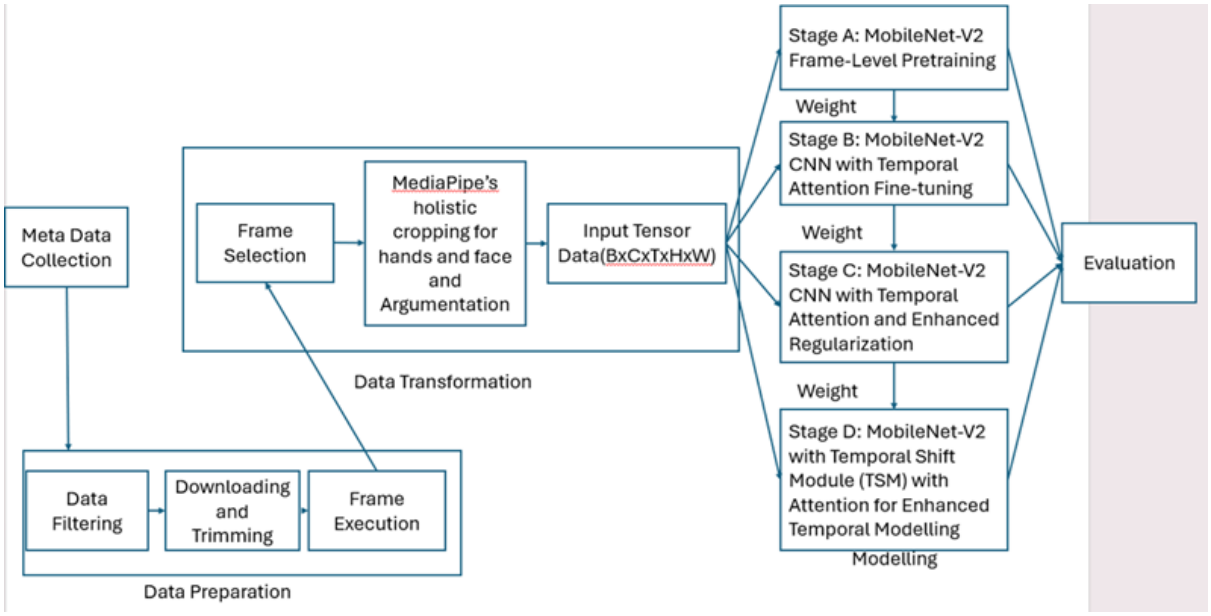

Figure 1: KDD Methodology

paper where this dataset is described and different models were evaluated (Vaezi Joze and Koller; 2019). This dataset in json format and consists of over 25000 videos' YouTube link with more metadata for 1000 distinct classes. Each class in the dataset represents a single word.

3.2 Data Preparation

3.2.1 Data Filtering

The data for all 1000 classes was filtered down to the first 100 classes (MS-ASL100) from all train, test and validation json file and created separate json file for 100 classes for train, test and validation.

3.2.2 Downloading and Trimming

Using filtered json files I downloaded videos using the URLs for that 'yt-dlp' library was used and then trimmed the videos using the start and end time given in the respective json files. 'ffmpeg' library was used for trimming and these videos were kept separately from the raw videos.

3.2.3 Frame Extraction

I extracted frames from the trimmed videos using cv2 library so that while training model does not need to extract frames on the fly memory which makes training period longer. For extracting frames in sequences, 16 fps (frame per second) was maintained (Figure 2). These extracted frames were saved in a structured folder path so that it can be easily understandable.

Figure 2: Extracted sequential frames from a video

3.2.4 MediaPipe-Holistic Cropping

Every frame underwent through cropping process using mediapipe holistic for landmarks detection before feeding to the model. For cropping, hand was given priority first and the face for stage B, C, D (Figure 3), where for stage A, priority was given on both hand and face.

Figure 3: Priority-based cropping

3.3 Modelling

3.3.1 Stage A: MobileNet-V2 Frame-Level Pretraining

In first stage (stage A) a 2D convolution neural network MobileNet-v2 was trained using the extracted frames. This stage's training helps the model to get familiar with the frames, such as hands, face etc. After reviewing previous research work, importance of landmarks detection has been realized and used in our method also 2.4(Palaniappan et al.; 2025). Every frame underwent through cropping using mediapipe holistic for landmark detection and resized to 160×160 pixels and augmented using random horizontal flipping and color jittering techniques. MobileNet-V2, which is pretrained on ImageNet dataset, was fine-tuned using AdamW optimizer with a learning rate of 1e-4 over ten epochs and saved the

model weights. This weight basically provided foundational visual representations used in subsequent stages.

3.3.2 Stage B: CNN with Temporal Attention Fine-tuning

I expanded the model here in stage B by integrating a temporal attention mechanism, which makes the model prioritize frames based on temporal importance. Input data sequences were sampled uniformly across 32-frame clips, using an effective cropping approach prioritizing hands first then the face, with 60-pixel padding. Stage A's backbone was used in this stage of model training. This stage's model training involved a two-phase approach where initial backbone were freeze followed by gradual unfreezing. AdamW optimizer was utilized, alongside a class balanced sampling method and a mini-batch Mix-Up augmentation strategy. Thus, after training, the weight was saved so that next stage's model (stage C) can load this weight as backbone weight. Also training performance was monitored through validation accuracy and loss metrics for the evaluation.

3.3.3 Stage C: CNN with Temporal Attention and Enhanced Regularization

In stage C, we built on stage B model by loading its trained weights as initialization which helped model not to learn from scratch, instead it can start from already learned representation. Also in this stage, stronger regularization techniques were applied compared to stage B so that model can learn better and reduce overfitting. This included more aggressive data argumentation techniques such as random resized cropping, Gaussian blur and random erasing, to help the model with learning more different variations in lighting, background and gesture appearance. A warm-up training phase were initiated with the backbone layers, which was retained from stage B, remained frozen at beginning After completing the warm-up phase, the backbone layers got unfrozen and entire network was finetuned using a lower learning rate so that it can maintain stability while training. These newly added techniques with temporal attention module which already present in stage B, provides a significant improvement in accuracy compared to previous stage.

3.3.4 Stage D: Integration of Temporal Shift Module (TSM) with Attention

In the stage D, with loading the training weight of stage C, a new layer of Temporal Shift Module (TSM) was introduced in the model's architecture with existing temporal attention. This addition allowed the network to capture short-term temporal relations between frames. This time the sequenced length was increased to 64 frames so that the model could observe the gestures more perfectly. Basically TSM rearranges certain feature channels across the neighboring frames, which can implicitly add motion information into the feature representation. Everything else is same as stage C except the newly added TSM. Even the training procedure also followed the same procedure as stage C. It included strong argumentation and regularization strategies with learning rate scheduling. After training was completed, the final weight of the model was saved, and the full training history was recorded for evaluation.

3.4 Evaluation

The models developed at each stage were evaluated using validation datasets. For all stages the core metrics were training accuracy, training loss, validation accuracy and

validation loss. When the model was developed that time training history was saved as well. Using those 'npy' files, each stages's model was evaluated. And also separate MS-ASL test dataset was used to evaluated model's performance. For the stage d's model additional evaluation metrices like macro average and weighted average precision, recall, and F1-score were used to check model's ultimate performance using test dataset. This systematic evaluation stage by stage helps to understand the performance gain in each stage by doing architectural changes or training strategies.

4 Design Specification

In this section, the design of the network architectures of the proposed American Sign Language(ASL) recognition models are illustrated and explained in detail. For making this project I developed four different models in four stages. Throughout the all four stages unmodified 2D CNN MobileNet-V2 feature extractor (inverted-residual blocks with depth-wise separable Convolutions) were used. In stage A, we trained our model using the same MobileNet-V2 where ImageNet weights were loaded to the network. After training the resulting checkpoint was then transferred to stage B and B's weight to C and C's weight to D. MobileNet-V2 layers details can be viewed here in this paper Sandler et al. (2018). Before processing with the model's network architecture and process flow, data preparation and transformation process for feeding to the model, would be discussed first.

4.1 Input data preparation and Transformation

For Stage A (frame-level pre-training) each frame is first cropped using mediapipe holistic which will also be followed other stages. A square crop with 40 pixels of padding is extracted, resized to 160 $\times$ 160 pixels, converted to a tensor, and normalized using ImageNet mean and standard deviation values. No temporal stacking is done here as every frame is treated individually, so the model input is a 4D tensor with shape (B $\times$ 3 $\times$ 160 $\times$ 160), where B is the batch size. During training argumentation was applied but not that heavy as stage B,C,D. Random horizontal flipping and slight color jitter to help the model generalize to variations in lighting and appearance.
For Stage B,C,D the process of data preparation for model feeding starts with decomposing each video into RGB frame. Mediapipe Holistic cropping strategy(3.2.4) and 60 pixels padding are applied to each frame. After that each frame is resized to 160x160 pixels and transformed using ToTensor() followed by ImageNet and Standard Deviation Normalization.
Stage B Argumentation : Random horizontal flip and color jitter.
Stage C Argumentation: Including Stage B's augmentations more stronger regularization with random resized crop, Gaussian blur and random erasing were added.
Stage D Argumentation: Same as Stage C, But adapted longer sequences means number of time frames (T) was given 64. For stage B and C, it was 32.
After transformation, frames are stacked to form a 5D tensor (B $\times$ 3 $\times$ T $\times$ 160 $\times$ 160), where T is the number of frames per clip where 32 for Stages B, C, and 64 for Stage D. C =3 means number of channels in this case, it is RGB channels with RED, GREEN, BLUE channels. This stacked tensor is then passed into the backbone network for feature extraction.

4.2 Models Architecture with Process flow

4.2.1 Stage A: Stage A: MobileNet-V2 Frame-Level Pretraining (Warmup)

Stage A's model architecture is simple. 4D tensor (B × 3 × 160 × 160) is sent one by one through an unaltered MobilenetV2 network which loaded with ImageNet weight. The backbone contains 17 inverted residual blocks, which gradually reduce the spatial resolution down to 5 × 5 feature map with 1280 channels. A global average pooling layer then compresses this map into a single 1280 dimensional feature vector. The original ImageNet classifier is replaced with a newly initialized fully connected layer that maps 1280 to 100 to match the number of sign classes in our dataset. The output logits are passed to the softmax to generate class probabilities. This model architecture does not do any temporal modelling as model focuses completely on learning to recognize static hand shapes and contextual visual features from single frames. It can be consider as a warm-up phase while keeping the backbone structure unchanged.

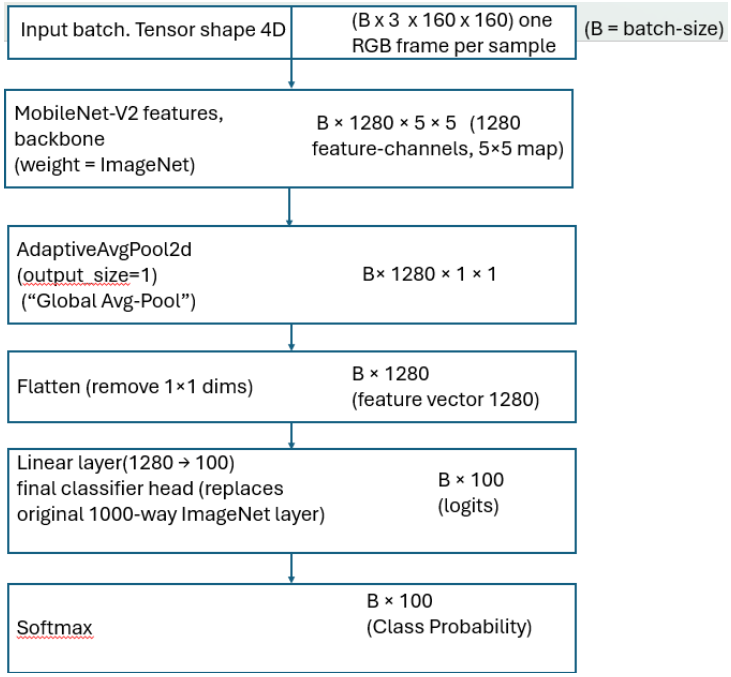

Figure 4: MobileNet-V2 Frame-Level Pretraining

4.2.2 Stage B: Mobilenet-V2 CNN with Temporal Attention Fine-tuning (5)

This time in this stage, model was extended to handle short video clips instead of individual frames. 5D tensor (B × 3 × T × 160 × 160) is passed through the model, where B is the batch size and T is the number of frames (32).

1. **Reshape for frame-level processing**
   The time dimension (T) was merged into the batch so that the MobileNet-V2 backbone could process each frame independently in parallel. This transformed the input into (B.T × 3 × 160 × 160), which means 128 images, if B=4 and T=32.

2. **Feature extraction with MobileNet-V2**
   The backbone was initialized with Stage A's trained weights, so the model can

start with a strong static handshape recognition capability and frames that passed through MobileNet-V2 produced 1280 channel feature with map size (5 × 5).

3. **Global average pooling and projection**
   A global average pooling layer (adaptive average pool) compressed each feature map to a single 1280 dimensional vector. This was followed by a linear projection (1280 to 256) with Batch Normalization and ReLU activation, producing a compact 256 dimensional embedding for each frame.

4. **Temporal attention mechanism**
   The embeddings were reshaped back to (B × T × 256) so that the temporal attention module can operate with frames. A linear layer (256 to 1) computed an attention score for each frame and a softmax function normalised these scores with frames so that they summed to 1 for each video.

5. **Weighted aggregation**
   The attention weights were for producing a single 256 dimensional vector that summarised the most informative moments of the clip.

6. **Final classification**
   Another linear layer was used and produced logits for the 100 sign classes, which were later converted to probabilities using softmax function.

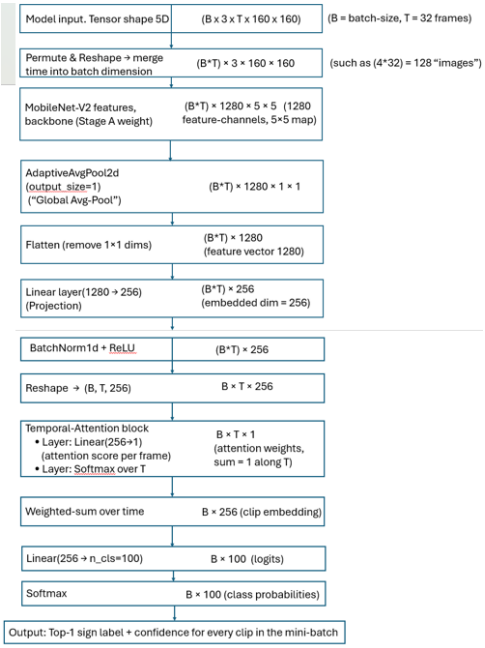

Figure 5: Mobilenet-V2 CNN with Temporal Attention

4.2.3 Stage C: Mobilenet-V2 CNN with Temporal Attention and Enhanced Regularization (Figure: 6)

This time, the model was loaded with the training weights of Stage B instead of starting from scratch. This allowed the network to retain the temporal attention learning from

Stage B. All of the layers of this stage (stage C) is same as Stage B except the Droupout layer with given rate of 0.5. This was added after the BatchNorm + ReLU step. This basically helped model to prevent overfitting by randomly deactivating neurons during training.

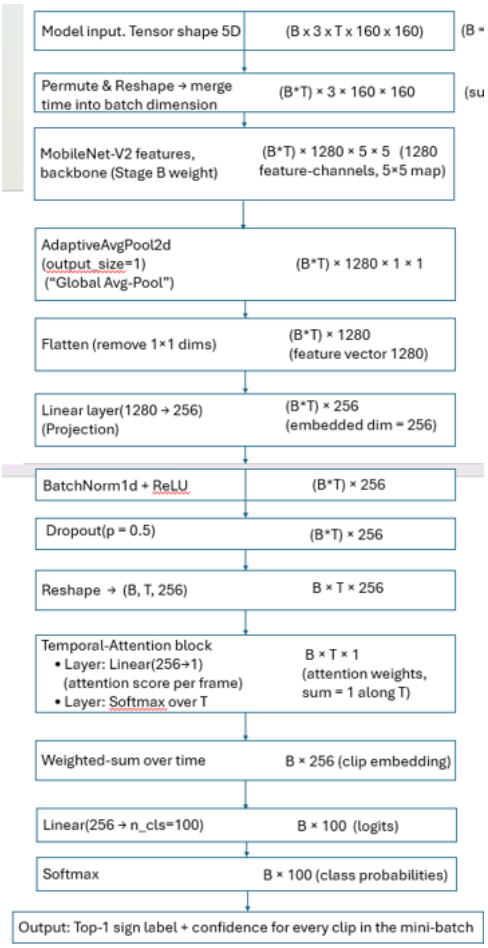

Figure 6: Mobilenet-V2 CNN with Temporal Attention and Dropout

4.2.4 Stage D: Mobilenet-V2 CNN with Temporal Shift Module (TSM)) with attention (Figure: 7)

This time, the model was loaded with the training weights of Stage C instead of starting from scratch. This basically allowed the network to retain both the temporal attention learning and the regularization's benefits from Stage C. All of the layers in this stage were the same as Stage C, except the Temporal Shift Module (TSM), which was added right before the backbone network. TSM rearranged a fraction of the feature channels across adjacent frames by shifting 1/8 to the previous frame, 1/8 to the next frame and keeping the remaining 3/4 in place. With this added technique, model can capture short-term temporal motion without adding extra parameters which enhance the sign recognizing function. Except that other layer of the network is as like as stage C's model.

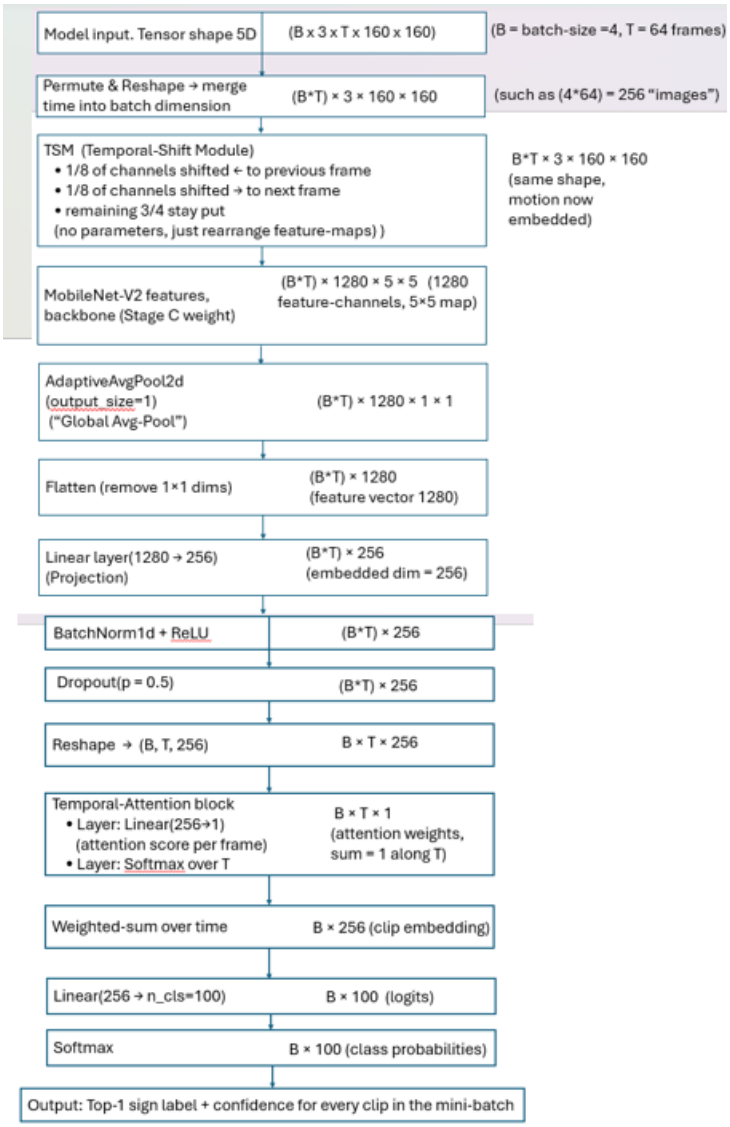

Figure 7: Mobilenet-V2 CNN with TSM + Attention

5 Implementation

This section describes the practical implementation of the proposed system by mentioning and explaining the tools, environment and procedures used to transform the design into a working solution. Architectural details and data preparation process had already been described in Sections 3 and 4.1. Here in this section, we focused on the building final system for training and evaluation.

5.1 Tools, Libraries and Environment Used

This project was developed in Python 3.12 in 'PyCharm' IDE, using 'PyTorch' and 'TorchVision' as the main deep learning framework. For working with this project different libraries were used such as, 'NumPy' for numerical operations , 'OpenCV' for video and image input and output, 'MediaPipe' for landmark detection and cropping, 'Streamlit' for building the real-time demo interface, and 'tqdm' for progress tracking. Raw MS-ASL100 videos were downloaded using 'yt_dlp' and trimmed with 'FFmpeg' and also did frame extraction. Image processing was also supported by the Pillow (PIL) library. For evaluation, 'scikit-learn' used for classification metrics. Data augmentation leveraged 'TorchVision' transforms, including random resized cropping, horizontal flipping, colour jittering, Gaussian blur, and random erasing with MixUp applied in later stages.

5.2 Hardware Used

All training and evaluation experiments were done by using my laptop which has a 13th Gen Intel Core i7-13620H CPU, 16 GB RAM, and an NVIDIA GeForce RTX 4050 GPU (6 GB). This GPU was used for training model and evaluation as we work with sequence of frames or videos. Using this laptop configuration, I trained each stage of the model to be efficiently without requiring cloud-based resources, though for training each stage's models all together took around 87 hours where stage D's model itself took around 57 hours.

5.3 Training Procedure and Hyper-parameters

5.3.1 Hyperparameters used during training

Table 2: Stage-wise hyperparameters used during training

Training Parameter	**Stage A**	**Stage B**	**Stage C**	**Stage D**
Frames per clip T	1	32	32	64
Alpha_mix (Mix-Up coefficient)	0	0.1	0.1	0.1
E_freeze(Frozen epochs)	0	5	5	3
LR_backbone_frozen	1e-4	1e-4	1e-4	1e-5
LR_backbone_unfrozen	1e-4	2e-4	1e-4	3e-5
LR_head	1e-4	5e-4	5e-4	3e-4
Patience for early-stop	-	-	7	7
Number of Epoch_head	10	30	50	50

5.3.2 Batch training loop algorithm for each stage

1. Create or Build PyTorch Model that loaded the Network for each stage such as :
   - Stage A: Load MobileNet-V2
   - Stage B: Load same backbone with Temporal-Attention head
   - Stage C: Load Stage B architecture including a Dropout layer
   - Stage D: Load Stage C architecture including a Temporal-Shift Module (TSM)
2. Create parameter groups
   - group_backbone = every layer that came from MobileNet-V2
   - group_head = all added new layers (projection, attention, TSM, etc.)
   - Give group_backbone the learning-rate `LR_backbone_frozen` and group_head the learning-rate `LR_head`.
   - Set `requires_grad = False` for group_backbone so it really is frozen during the first `E_freeze` epochs.
3. Instantiate the optimiser and scheduler
   - AdamW optimiser with weight decay 1×10^{-4}
   - Cosine annealing learning-rate schedule that ends at 1×10^{-6}
4. Initialise helpers
   - `best_val_loss = very_large_number`, `no_improvement = 0` and `history = empty`
5. Training epochs

   For epoch from 1 to `E_max`:

   (a) If `epoch == E_freeze + 1`
      - unfreeze group_backbone (`requires_grad = True`)
      - change its learning-rate to `LR_backbone_unfrozen`

   (b) Loop over every mini-batch (clips, labels)
      i. If `alpha_mix > 0` apply Mix-Up to the whole clip tensor
      ii. Forward pass through model outputs logits
      iii. Compute cross-entropy loss (label-smoothing 0.1)
      iv. Update the model weights: back-propagate the loss through the network and apply the optimiser to adjust all trainable parameters.

   (c) Advance the learning-rate schedule: move the cosine schedule forward by one position so the next epoch uses the newly lowered learning rate.

   (d) Run a full validation pass; record `val_loss` and `val_top1`.

   (e) Append the epoch's training and validation metrics to history.

   (f) Model's weight selection
      - If `val_loss` is lower than `best_val_loss`

- store the current weights as `best_model_weights`
- set `best_val_loss = val_loss` and `no_improvement = 0`

• Else
- increase `no_improvement` by 1

(g) If `no_improvement` is now equal to `patience` then break (early-stop).

6. Return the best weights and the complete metric history.

5.3.3 Optimizer and Scheduler

For all training stages, the 'AdamW' optimizer was used with two different parameter groups, such as one for the backbone and another for the classification head (newly added layers). Basically, training began with the backbone frozen, so that only the classification head layers could get updated. Once the backbone was unfrozen, its learning rate was set between 20% to 30% of the head's learning rate to avoid catastrophic forgetting of the features which it had already learned. A cosine annealing scheduler was also applied to both groups, which gradually reduced learning rates to zero. This smooth decay technique was used so that it can reduce learning rates automatically while training.

5.4 Test Pipeline for Live-Webcam Uploaded Videos

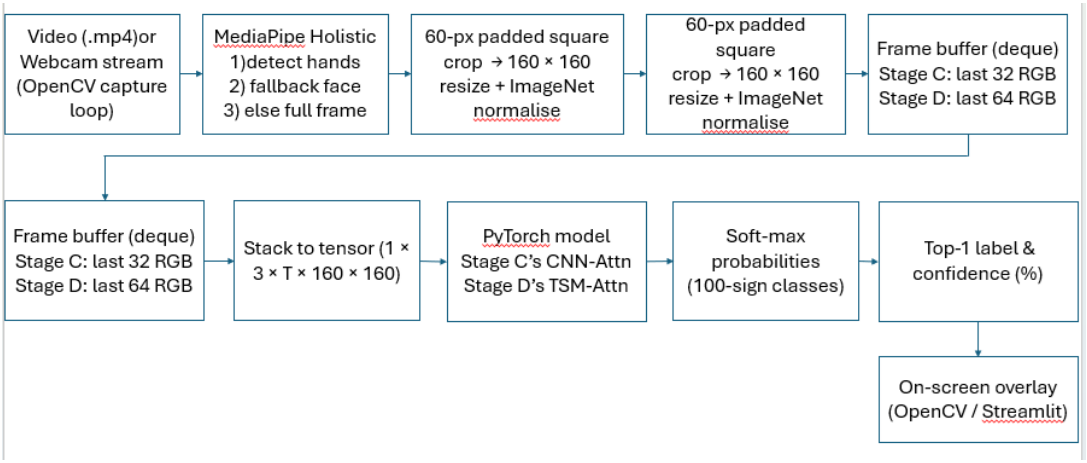

Figure 8: Test Pipeline for Live Webcam and Uploaded Videos

The figure: 8 describes the pipeline or workflow for live video classification using web camera or using pre-recorded videos. So frames are received either from the webcam or from an uploaded video. Each frame is first processed by MediaPipe Holistic, which tries to detect the hands. If hands are not found, it falls back to the face, and if that also fails, it uses the full frame. It is similar like early described input data processing and transformation(4.1). The selected area is cropped, resized to 160 × 160 pixels, and normalized using ImageNet statistics. The processed frames are then stored in a sliding buffer where 32 frames is selected for Stage C (CNN + Attention) or 64 frames for Stage D (TSM + Attention). When the buffer is full, the whole clip 5D tensor (1 × 3 × T × 160 × 160) is sent through the chosen model. The model produces softmax probabilities for all 100 ASL signs. The top prediction and its confidence are shown live on the video

stream using OpenCV or returned through Streamlit when testing uploaded clips. This completes the inference process, which is later used for the practical demonstration and evaluation.

6 Evaluation

The evaluation of our American Sign Language recognition system was conducted for four stage's model performance. The main metrics used for the evaluation were Accuracy and Loss. For each stage the model trained on different numbers of epochs such as, for stage A the number is 10, stage B it is 30 where C and D is 40 (early stopping occurred). The training and validation performance for accuracy and loss for all stages are shown in the figure:9.

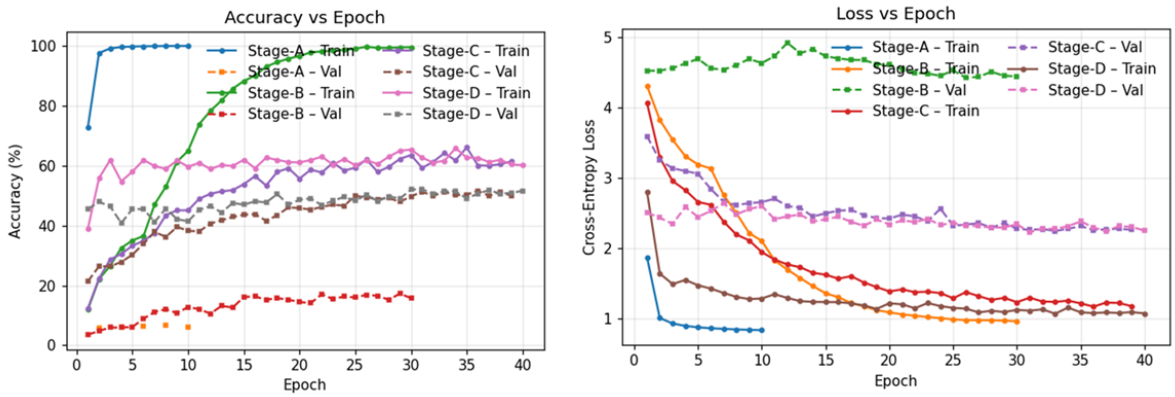

Figure 9: Accuracy and Loss

6.1 Experiment with MobileNet-V2 Frame-Level Pretraining (stage A)

Stage A basically worked as an initial warm up phase, focusing on frame level training using the MobileNet-V2. Over the 10 epochs, training accuracy increased steadily and reached approximately 99%, while training loss decreased to below 1%, indicating model's learning ability to identify basic visual patterns, such as getting familiar with hands, facial structures etc. However, validation accuracy remained low, which is around 7%. The outcome was expected as the primary goal of Stage A was not to achieve high generalization. These learned features were used for stage B by loading the weight of stage A's model, where the temporal modeling was introduced.

6.2 Experiment with Mobilenet-V2 CNN with Temporal Attention Fine-tuning (stage B)

From the Stage B, the model started learning the actual classification task as temporal attention was given. From the figure no: 9, a consistent rise in training accuracy up to 95% was observed. While validation accuracy improved slightly but remained below 18%, indicating overfitting. The loss graph also indicating that model was not learning.

Where the training loss drops significantly, the validation loss remains nearly flat, indicating limited generalization. The test accuracy came 13.46% using test dataset, which confirmed the above state of overfitting and not generalizing (figure: 10).

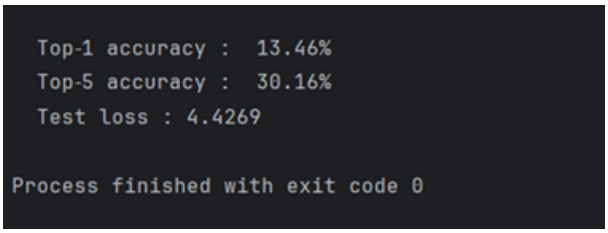

Figure 10: Accuracy

6.3 Experiment with Mobilenet-V2 CNN with Temporal Attention and Enhanced Regularization (Stage C)

Here the test accuracy recorded using MS-ASL test data for stage C's model is 49.42% (figure: 11) for top-1 accuracy. For the top-5 it is 74.71%, where the average test loss is 2.1824. This model performs better than stage B's model. Here in the stages, I made argumentation and regularization more strong and used dropout and obviously used the training weight of stage B here which we achieved a good score. The training accuracy increased to around 65%, where the validation accuracy gradually reached close to 50%, showing a better generalization trend compared to Stage B (figure: 9). The loss graph for Stage C shows a slow but steady decline in both training and validation loss, indicating model generalization and early stopping occurred at the 40th epoch as there were no improvements for 7th continuous epoch, which helped avoid overfitting. I have also shown a live classification or prediction for the 'boy' sign using live web camera (figure no: 11).

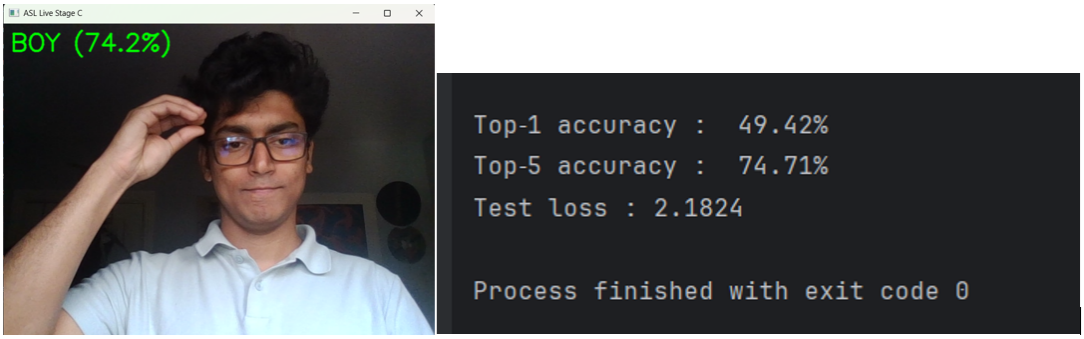

Figure 11: Experiment with Mobilenet-V2 CNN with Temporal Attention and Enhanced Regularization

6.4 Experiment with Mobilenet-V2 CNN with Temporal Shift Module and Attention (Stage D)

In Stage D, where the model operates with Temporal Shift Module (TSM) along with attention, by observing the graph (figure no: 9) carefully it can be said that the training accuracy remained stable (60–65%) and validation accuracy keep fluctuating around 53%. The gap between training and validation accuracy was smaller than Stage B, which suggests that the model with TSM and attention generalizes better. Additionally, both training and validation losses were stable and low, indicating lower chances of overfitting. The training also stopped early at 40th epoch, which helped model not to get overfitted. Overall accuracy using test data is 51.97% for top-1 and 77.49% top-5, where average loss was 2.1122. All these numbers indicate better performance than stage C. Also, a demo live presentation has been given where it was able to detect 'BAD' word by capturing movements properly (figure no: 12).


```
Test set : 431 videos   100 classes

Average loss    : 2.1122
Top-1 accuracy      :  51.97%
Top-5 accuracy      :  77.49%

Precision / Recall / F1 (Macro & Weighted Avg):
Macro  Avg Precision: 0.560   Recall: 0.559   F1-score: 0.508
Weighted Avg  Precision: 0.588   Recall: 0.520   F1-score: 0.501

Process finished with exit code 0
```

Figure 12: Experiment with Mobilenet-V2 CNN with Temporal Shift Module and Attention

6.5 Discussion

From Table 3, it is clearly noticeable that the technique we followed throughout the project was effective, as models achieved better results at each stage. Finally, in the last stage, Model with TSM including attention achieved 51.97% accuracy for top-1 where 77.49% for top-5. This improvement is primarily the result of using weight transfer from

Table 3: Models Performance on each stage

Model	**Epoch**	**Training Time (Approximate Hours)**	**Accuracy**
MobileNet-V2(frame warmup)	10	5	7%9
Mobilenet-V2 CNN with Temporal Attention	30	9	13.46%
Mobilenet-V2 CNN with Temporal Attention and Enhanced Regularization	40(early stop)	16	49.42%
Mobilenet-V2 CNN with TSM and Attention	40(early stop)	57	51.97%

Table 4: Comparison on ASL 100 words(top-1 accuracy).

Authors & Year	**Best-Performing Model**	**Dataset**	**Accuracy**
Li et al. (2025) (Li et al.; 2025)	Uni-SIGN (RGB + pose + gloss)	MS-ASL100	93.79%
Zuo et al. (2023) (Zuo et al.; 2023)	VKNet + language aware stream	MS-ASL100	90.49%
Izutov (2020) (Izutov; 2020)	S3D-MobileNetV3 + ST attention	MS-ASL100	85.00%
Maruyama et al. (2024) (Maruyama et al.; 2024)	Multi-stream MSNN	MS-ASL100	83.90%
(Kumari and Anand; 2024)	MobileNetV2 + attention+LSTM	WLASL100	84.65%
Vaezi Joze and Koller (2019) (Vaezi Joze and Koller; 2019)	I3D	MS-ASL100	81.76%
(Li et al.; 2020)	I3D	WLASL100	65.89%
Me (2025)	**Proposed: 2-D CNN MobileNetV2 + Attention + TSM**	**MS-ASL100**	**51.97%**

one stage to another, along with TSM, attention and augmentation techniques. From table 4 It can be said that our proposed model's accuracy is lower than the previous work. This gap was expected as our goal was to make a light model using 2D CNN with TSM+ attention where we are successful but the accuracy we achieved is lower than the previous paper as we can see. If you compare the model's weight like heavy computational architecture models such as I3D and Transformers, our model is working well being lightweight. It can nicely classify live signs as well with temporal movement (figure: 12) . It can be said that our proposed model is performing well but not as effective as heavy models like 3D CNNs and Transformers as these models have 3D kernel that can detect the motion effectively where we worked with only 2D Kernel. Despite having these limitations in our proposed model, our accuracy is reasonably good. Another reason for not getting more accuracy could be the unavailability of data. I was supposed to train models using 5,736 videos, where only 2130 downloadable data was available. For which reason our model lacked more features to learn, which could be a reason for not getting accuracy above 60%. Because of missing data, the classes are imbalanced. Though WeightedRandomSampler technique was applied to mitigate this by giving more attention to less classes, it would not be enough if imbalance is huge. This could be another reason for the lower accuracy.

7 Conclusion and Future Work

7.1 Conclusion

My study set out to answer the research question:

"Can a lightweight 2D CNN with Temporal Shift Modules (TSM) and attention, achieve sign-language recognition performance comparable to heavyweight 3D CNN and transformer models? "

This study implemented a successful lightweight weight model using 2D CNN MobileNet-V2 with Temporal Shift Module and Attention which achieved 51.97% Top-1 and 77.49% Top-5 accuracy on the MS-ASL100 dataset with limited training data (only 2130 usable videos out of 5736).

Though results of my proposed proposed model indicate that while 2D CNN with Temporal Attention and TSM can deliver reasonable accuracy for temporal movement and run efficiently, it cannot match the performance of heavy 3D CNNs or Transformer architectures in motion classifying tasks due to their superior temporal modeling capabilities.

7.2 Future Work

A proposal of future work is given below, by following them we can gain better performance while keeping the idea of lightweight model intact.

- Combine both MS-ASL with WLASL datasets to get more diverse data for training so that models can learn more diverse features.
- Use checkpoints which were trained on large-scale action recognition datasets such as Kinetics 400 or 700.

- Explore longer clips (such as 96 frames) and increase input resolution from 160px to 224px to capture more details.
- Use advanced regularization and training techniques such as:
  - Apply CutMix or RandAugment for stronger data augmentation.
  - Implement curriculum sampling to gradually increase complexity.
  - Optimize label smoothing and use gradient clipping for stability.
  - Adopt cosine learning rate schedules and discriminative learning rates (different LR for backbone and head).
- Keeping core idea of lightweight model (MobileNet-V2) enhance Temporal Modeling such:
  - Add LSTM (RNN) layers on top of TSM for better sequence learning or
  - Replace TSM with a lightweight transformer like 'MobileViT' for improved global temporal attention.

References

Babu, A. R., Himaja, Y. S., Stuthi, S. J., Mamatha, B. et al. (2024). Transformation of sign language to text in digital era using deep neural network, *2024 IEEE 16th International Conference on Computational Intelligence and Communication Networks (CICN)*, IEEE, pp. 1278–1283.

Chophuk, P., Chamnongthai, K. and Chinnasarn, K. (2022). Backhand-approach-based american sign language words recognition using spatial-temporal body parts and hand relationship patterns, *Sensors* **22**(12): 4554.

Izutov, E. (2020). Asl recognition with metric-learning based lightweight network, *arXiv preprint arXiv:2004.05054* .

Kumari, D. and Anand, R. S. (2024). Isolated video-based sign language recognition using a hybrid cnn-lstm framework based on attention mechanism, *Electronics* **13**(7): 1229.

Li, D., Rodriguez, C., Yu, X. and Li, H. (2020). Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison, *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 1459–1469.

Li, Z., Zhou, W., Zhao, W., Wu, K., Hu, H. and Li, H. (2025). Uni-sign: Toward unified sign language understanding at scale, *arXiv preprint arXiv:2501.15187* .

Lin, J., Gan, C. and Han, S. (2019). Tsm: Temporal shift module for efficient video understanding, *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7083–7093.

Maruyama, M., Singh, S., Inoue, K., Roy, P. P., Iwamura, M. and Yoshioka, M. (2024). Word-level sign language recognition with multi-stream neural networks focusing on local regions and skeletal information, *IEEE Access* .

Mudumbai, K., Karthik, M., Shalini, N. and Trisha, N. (2024). Sign language recognizer using convolutional neural networks, *2024 2nd World Conference on Communication & Computing (WCONF)*, IEEE, pp. 1–7.

Palaniappan, R., Rishitha, C., Nikhil, M. S. and Pradeep, B. (2025). A novel explainable deep learning model for sign language recognition and translation, *2025 6th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pp. 1045–1052.

Prachi Rawat, P. K., Tamta, V. K. and Kumar, A. (2025). A comprehensive approach to indian sign language recognition: Leveraging lstm and mediapipe holistic for dynamic and static hand gesture recognition, *EAI Endorsed Transactions on AI and Robotics* **4**.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520.

Tadele, B., Assabie, Y. and Tegegne, T. (2025). Word level ethiopian sign language recognition using hybrid cnn-lstm model, *2025 3rd International Conference on Advancement in Computation & Computer Technologies (InCACCT)*, IEEE, pp. i–v.

Vaezi Joze, H. and Koller, O. (2019). Ms-asl: A large-scale data set and benchmark for understanding american sign language, *The British Machine Vision Conference (BMVC)*.
URL: *https://www.microsoft.com/en-us/research/publication/ms-asl-a-large-scale-data-set-and-benchmark-for-understanding-american-sign-language/*

Wang, Z. and Yang, X. (2025). An efficient 3d convolutional neural network with channel-wise, spatial-grouped, and temporal convolutions, *arXiv preprint arXiv:2503.00796* .

Zhou, W., Zhao, W., Hu, H., Li, Z. and Li, H. (2024). Scaling up multimodal pre-training for sign language understanding, *arXiv preprint arXiv:2408.08544* .

Zuo, R., Wei, F. and Mak, B. (2023). Natural language-assisted sign language recognition, *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14890–14900.