IEEE 2nd International Conference on Computing, Applications and Systems (COMPAS 2025)
23–24 October 2025, Kushtia, Bangladesh

SignNet-Nano: Efficient Sign Language Recognition for Real-Time Edge Deployment

Rifat Zabin*, Md. Riasat Tanjim Hossain‡, Khandaker Foysal Haque§, K M Rumman§
* Department of Computer Science Engineering, BRAC University, Bangladesh
‡ Department of Electrical and Electronic Engineering, Islamic University of Technology, Dhaka, Bangladesh
§ IEEE Graduate Student Member

Abstract*—Real-time american sign language (ASL) alphabet recognition can enable inclusive communication on next-generation edge devices like virtual reality (VR) headsets and smart glasses. However, existing models are often too computationally demanding for such platforms, resulting in fast battery depletion and degraded performance. To address this challenge, we introduce `SignNet-Nano`, an ultra-lightweight convolutional neural network (CNN), tailored specifically for efficient ASL alphabet recognition on constrained edge hardware. The proposed model integrates depthwise separable (DS) convolutions, squeeze-and-excitation (SE) attention, and global average pooling (GAP) to achieve high accuracy with a compact architecture comprising only $< 20K$ parameters and $< 0.1MB$ model size. Unlike prior work, `SignNet-Nano` is explicitly designed with deployment efficiency as a primary goal, enabling real-time inference with minimal performance degradation while requiring significantly less energy, latency, computation, and memory usage than state-of-the-art (SOTA) lightweight edge models. We profile inference performance in terms of Floating Point Operations (FLOPs), inference latency, frames per second (FPS), and memory footprint on a diverse set of platforms, including three edge devices— Jetson Nano, Jetson Xavier NX, and Raspberry Pi 4— as well as three high-performance systems— Apple Mac M3 (MPS backend), NVIDIA RTX 4070 GPU, and an Intel Core i7 12th Gen CPU. Experimental results show that `SignNet-Nano` achieves classification accuracy within 1% of the best-performing baseline while reducing inference time (i.e., latency) and FLOPs by up to 73.7% and 97.1%, respectively, while improving the energy efficiency by up to 79.3%.

I. Introduction

The ability to recognize american sign language (ASL) alphabet gestures in real time has the potential to significantly enhance inclusive communication [1], [2], particularly for the deaf and hard-of-hearing community. This is increasingly relevant in the context of next-generation edge-centric platforms such as augmented reality (AR) and virtual reality (VR) headsets, smart glasses, and wearable devices [3], [4], where real-time gesture-based input can serve as an intuitive interaction modality. However, these devices are still quite limited in terms of the performance delivered to the end user. Studies indicate that achieving truly immersive 360° video experiences requires at least 120 Hz frame rates and 8K resolution to avoid pixelation and mitigate motion sickness [5]. In contrast, most wireless VR headsets available today are limited to 4K resolution, with only a few models supporting frame rates in the 100–120 Hz range [6]. Moreover, these devices often lack the computational resources and power sources necessary to run complex deep neural network (DNN) workloads efficiently. As a result, deploying such models can lead to either rapid battery depletion or substantial degradation in application performance [7].

979-8-3315-5525-2/25/$31.00 © 2025 IEEE

Despite recent advances in deep learning-based sign language recognition, most existing models are designed with server-grade GPUs in mind. Architectures such as ResNet18 [8] and even lightweight alternatives like MobileNetV2 [9] remain too computationally demanding for real-time deployment on embedded platforms. Their high parameter count, high computational demand (i.e., high Floating Point Operations (FLOPs)), and poor latency performance contribute to thermal throttling, reduced battery life, and a sub-par user experience— ultimately limiting their practicality for edge-centric applications.

Thus, to address these challenges, we present `SignNet-Nano`, an ultra-lightweight convolutional neural network (CNN) architecture explicitly designed for real-time ASL alphabet recognition on resource-constrained devices. Our key design philosophy is to achieve high accuracy with minimal computational overhead. `SignNet-Nano` integrates depthwise separable (DS) convolutions, squeeze-and-excitation (SE) attention, and global average pooling (GAP) to maintain discriminative power while minimizing the required parameter count and FLOPs. To ensure practical deployability, we conduct a comprehensive evaluation across a diverse set of platforms, including three widely accessible edge devices— Jetson Nano, Jetson Xavier NX, and Raspberry Pi 4— as well as high-performance systems like an Apple MacBook—using metal performance shaders (MPS), NVIDIA RTX 4070 GPU, and Intel Core i7 CPU.

Summary of Contributions

- We propose `SignNet-Nano`, a compact CNN architecture with fewer than 20K parameters, featuring DS convolutions and SE attention modules for efficient ASL alphabet recognition.
- We present a unified deployment and benchmarking framework across heterogeneous devices, offering a realistic deployment performance comparison of `SignNet-Nano` with state-of-the-art (SOTA) approaches— ResNet18 [8] and MobileNetV2 [9].
- We demonstrate real-time inference capability of `SignNet-Nano` on three different edge devices— Jetson Nano, Raspberry Pi 4, and Jetson Xavier NX, along with other high-performing devices with significant improvements in frames per second (FPS), energy consumption, and inference latency compared to the SOTA models.

• To foster reproducibility and future research, we pledge to release our code base along with the benchmarking scripts publicly at https://github.com/rifatzabin/SignNet-Nano.

II. Related Works

Lightweight deep learning architectures have become a crucial scope of research in resource-constrained computer vision applications. EfficientNet, introduced by Tan and Le, employed compound scaling to optimize depth, width, and resolution, demonstrating strong performance-efficiency tradeoffs, making it suitable for vision tasks with limited computing resources [10]. He et al. proposed ResNet as a family of scalable models that balance precision and efficiency, leveraging residual connections that facilitate gradient flow in shallower networks [8]. MobileNetV2, proposed by Sandler et al., employs DS convolutions and inverted residuals with linear bottlenecks, making it highly suitable for mobile and embedded scenarios [9].

In the context of ASL recognition, Kasapbaşi et al. developed DeepASLR— a lightweight CNN tailored for ASL alphabet recognition. Their design focused on faster convergence and lower training complexity while achieving competitive accuracy [11]. Sharma and Singh proposed ASL-3DCNN, a depth image-based 3D CNN model to recognize ASL alphabets, utilizing spatiotemporal features to improve the accuracy of sign language classification [12]. Castro et al. proposed a sensor-free sign language recognition system using a multi-stream 3D CNN that fuses RGB frames, segmented hand and face regions, joint motion features, and generative adversarial network-generated depth maps. Their approach eliminates the need for depth sensors and demonstrates the effectiveness of multi-modal input for sign language recognition [13].

Regarding edge deployment, Sharma et al. explored the performance of a pose recognition model implemented on a Raspberry Pi 4, showing that with proper optimization, embedded inference is feasible even under tight memory and latency constraints [14]. Baciu et al. proposed "MLino Benc" [15], an open-source benchmarking tool that enables comprehensive evaluation of both classical machine learning models and neural networks on microcontroller-based edge devices. However, it lacks the support for power consumption analysis and remains limited to classification tasks.

Despite these advancements, the mentioned architectures consistently lack ultra-lightweight ASL models that are explicitly profiled on real-world embedded devices.

III. Under the Hood: Design and Training of SignNet-Nano

A. Model Architecture

To meet the dual objectives of deployment efficiency and classification accuracy, `SignNet-Nano` is designed as a compact CNN tailored specifically for edge devices. The architecture incorporates a lightweight feature extraction backbone based on DS convolutions, enhanced with SE attention modules, and concludes with GAP for compact prediction. The overall structure is illustrated in Fig. 1.

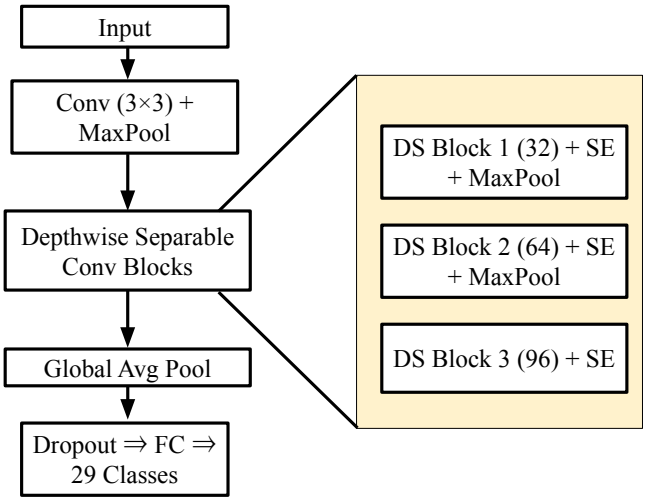


Fig. 1: Overview of the `SignNet-Nano` architecture.

The network processes $224 \times 224 \times 3$ input RGB images through an initial convolutional layer, followed by a stack of DS convolution blocks that progressively increase feature depth while reducing spatial dimensions. Each block is augmented with a SE attention module to enhance channel-wise feature selectivity. Finally, a GAP layer aggregates spatial information, and a fully connected output layer maps the latent features to the ASL alphabet classes. The following components describe the technical building blocks of this architecture in detail.

***DS Convolutions**:* DS convolutions offer a more efficient alternative to standard convolutional layers by decoupling spatial and cross-channel feature learning. In a standard convolution, spatial filters are applied across all input and output channels using a kernel of shape $K \times K \times C_{\text{in}} \times C_{\text{out}}$, where K is the spatial kernel size (e.g., 3 for a 3×3 filter), C_{in} is the number of input channels, and C_{out} is the number of output channels. In contrast, depthwise separable convolution factorizes this operation into two stages— (i) DS, which applies a separate $K \times K$ filter to each of the C_{in} input channels individually, and (ii) pointwise convolution, which uses $1 \times 1 \times C_{\text{in}} \times C_{\text{out}}$ filters to combine the resulting feature maps across channels. Mathematically, given an input feature map $\mathbf{X} \in \mathbb{R}^{H \times W \times C_{\text{in}}}$, where H and W represent the spatial height and width— the number of FLOPs required for a standard convolution is denoted by—

$$\text{FLOPs}_{\text{standard}} = H \cdot W \cdot K^2 \cdot C_{\text{in}} \cdot C_{\text{out}}. \tag{1}$$

Whereas, for a DS convolution, this reduces significantly due to the decoupled operations as—

$$\text{FLOPs}_{\text{DS}} = H \cdot W \cdot \left(K^2 \cdot C_{\text{in}} + C_{\text{in}} \cdot C_{\text{out}}\right), \tag{2}$$

demonstrating substantial computational savings, especially when K or C_{out} is large. This leads to a relative computational

saving of approximately $\frac{1}{C_{\text{out}}} + \frac{1}{K^2}$, which is significant in practical settings where $K = 3$ and $C_{\text{out}} \gg 1$. In `SignNet-Nano`, all convolutional layers beyond the initial input layer utilize this structure to substantially reduce inference cost while maintaining representational capacity.

GAP: To further reduce the number of trainable parameters and minimize the risk of overfitting, `SignNet-Nano` replaces traditional fully connected layers with a GAP layer prior to classification. GAP operates by aggregating spatial information across each feature channel, effectively reducing the spatial dimensions to a single scalar per channel. Formally, for a feature tensor $\mathbf{F} \in \mathbb{R}^{H \times W \times C_{in}}$, the GAP operation produces a C_{in} dimensional vector $\mathbf{z}$, by averaging each channel spatially as Equation 3.

$$z_c = \frac{1}{H \cdot W} \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbf{F}_{i,j,c}, \quad \text{for } c = 1, \ldots, C_{in} \tag{3}$$

This operation compresses the spatial structure while preserving the strength of feature activations in each channel, making it an effective replacement for dense layers in compact CNN architectures. Moreover, it enables a smoother transition into the final classification layer with minimal computational overhead.

SE Attention: To enhance the representational power of the lightweight convolutional backbone, each DS block in `SignNet-Nano` is augmented with a SE module. The SE block introduces a lightweight channel attention mechanism that adaptively recalibrates feature maps by modeling inter-channel dependencies. Given an intermediate feature map $\mathbf{U} \in \mathbb{R}^{H \times W \times C_{in}}$, the SE block performs three sequential operations—

`1) Squeeze:` Spatial information is aggregated using GAP to generate a channel descriptor vector as:

$$s_c = \frac{1}{H \cdot W} \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbf{U}_{i,j,c}, \quad \text{for } c = 1, \ldots, C_{in}, \tag{4}$$

where, s_c is the c-th component of the descriptor vector $\mathbf{s}$ and represents the average activation of channel c across all spatial locations.

`2) Excitation:` The descriptor vector $\mathbf{s}$ is passed through a two-layer fully connected network (a bottleneck structure) with a reduction ratio r, producing a channel-wise gating vector $\mathbf{e} \in \mathbb{R}^C$:

$$\mathbf{e} = \sigma\left(\mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1 \cdot \mathbf{s})\right) \tag{5}$$

Here, $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$ are trainable parameters and $\sigma(\cdot)$ denotes the sigmoid activation function applied element-wise.

`3) Scale:` The original feature map is reweighted channel-wise using the learned excitation vector, $\tilde{\mathbf{U}}_{i,j,c} = e_c \cdot \mathbf{U}_{i,j,c}$.

This gating mechanism allows the network to selectively amplify informative channels while suppressing less relevant ones, thereby improving discriminative capacity. In the context of ASL alphabet recognition, where many gestures differ by subtle features, this fine-grained channel emphasis contributes to the overall robustness of `SignNet-Nano`.

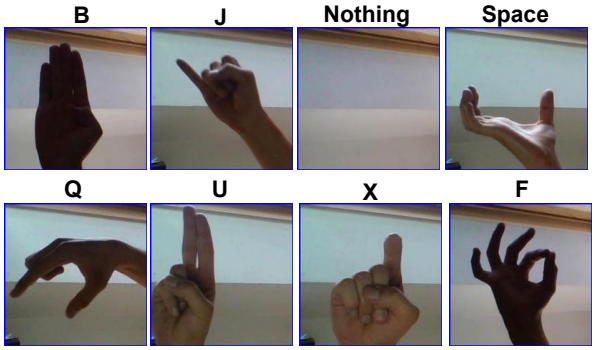


Fig. 2: Example images from the ASL Alphabet dataset showing hand gestures corresponding to various classes— B, J, Nothing, Space, Q, U, X, F.

B. Dataset and Preprocessing

We evaluate the proposed `SignNet-Nano` model using the publicly available Kaggle ASL Alphabet dataset[1] [16]. The dataset consists of 29 classes, covering the 26 letters of the English alphabet along with three additional classes: "nothing", "space", and "delete". Each class contains approximately $3,000$ RGB images of resolution 200×200 pixels, resulting in a total of over $87,000$ labeled samples. Example images from the dataset illustrating a variety of ASL gestures are shown in Fig. 2.

For training and validation, we randomly split the dataset into an 80% training set and a 20% validation set, ensuring a uniform distribution of samples across all classes. The split is performed only once and used consistently across all experiments to maintain fair comparison. To enhance generalization and reduce overfitting, we apply a set of data augmentation techniques during training. Each image is resized to 224×224 pixels and normalized to have zero mean and unit variance per channel. Augmentations include random horizontal flips, small rotations (up to $15°$), and color jittering to simulate real-world lighting variations. Furthermore, we apply *label smoothing* during training to prevent the model from becoming overly confident in its predictions, which improves robustness and generalization.

C. Training Strategy

To train the proposed `SignNet-Nano` model effectively while maintaining generalization and stability, we adopt a carefully tuned strategy combining robust optimization, regularization, and early termination. We use the categorical cross-entropy loss function with *label smoothing* to mitigate overfitting and reduce model overconfidence. Specifically, the one-hot target distribution is adjusted using a smoothing factor $\epsilon = 0.1$, such that the ground-truth class receives a

[1]https://www.kaggle.com/datasets/grassknoted/asl-alphabet

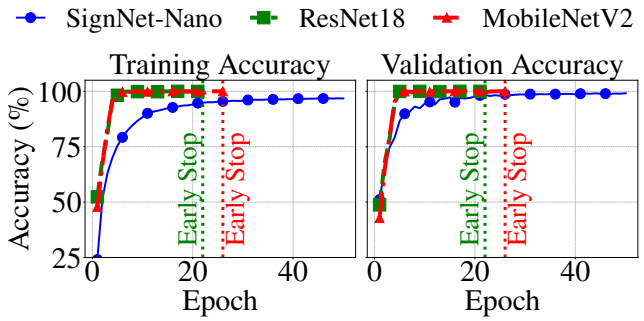


Fig. 3: Training and the validation accuracy of `SignNet-Nano` and SOTA baseline models. MobileNetV2 and ResNet18 converge faster, while `SignNet-Nano` maintains stable learning with competitive final accuracy.

label of $(1-\epsilon)$ while the remaining probability mass is evenly distributed among the other classes. This encourages the network to maintain predictive flexibility and has shown to improve performance, especially in multi-class classification tasks with subtle inter-class differences, such as ASL gestures. Optimization is carried out using the Adam optimizer with default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and an initial learning rate of 1×10^{-3}. To enhance convergence and avoid sharp minima, we employ a cosine annealing learning rate scheduler, which gradually reduces the learning rate to near zero as training progresses. This scheduling policy helps in achieving smoother convergence and better generalization during later stages of training. The model is trained using a batch size of 64 and monitored via validation loss. We apply *early stopping* with a patience of 5 epochs, halting training if the validation loss does not improve, which prevents overfitting and reduces unnecessary training time. Although all models converge within 35 to 45 epochs, both MobileNetV2 and ResNet18 exhibit faster convergence compared to `SignNet-Nano`, which learns more gradually but achieves stable and competitive accuracy, as illustrated in Fig. 3. All training is performed using PyTorch 2.0.1 on a workstation equipped with an NVIDIA RTX 4070 GPU (16 GB VRAM). The dataset is preprocessed on the go using the TorchVision data pipeline, and training is repeated across three different random seeds to ensure stability and reproducibility of the reported results.

IV. Real-World Deployment and Benchmarking

To assess the practical deployability of `SignNet-Nano`, we conduct real-world benchmarking across a diverse range of platforms, spanning both edge devices and high-performance computing systems. The goal is to evaluate the model's latency, throughput, and memory efficiency in realistic scenarios using a unified benchmarking script implemented in PyTorch.

The tested edge devices include the NVIDIA Jetson Nano, NVIDIA Jetson Xavier NX, and Raspberry Pi 4, which represent common low-power platforms suitable for on-device

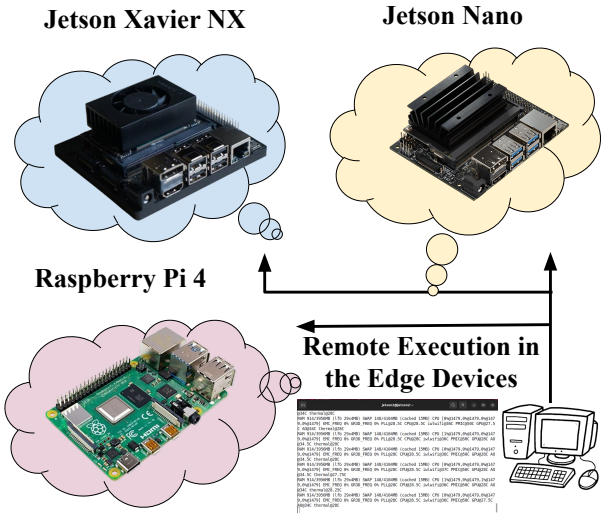


Fig. 4: Experimental setup for remote edge deployment at BRAC University, where all models, including `SignNet-Nano` are benchmarked on the edge testbed.

inference. In addition, we benchmark the models on three high-performance systems— an Apple MacBook powered by the M3 chip with MPS, an NVIDIA RTX 4070 GPU using the CUDA backend, and an Intel Core i7 12th Gen CPU. We execute inference remotely on these edge devices using benchmarking scripts to replicate real-world deployment, as depicted in Figure 4.

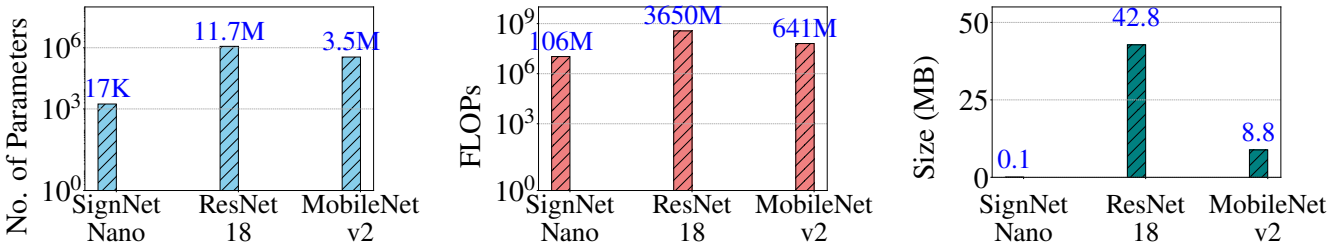
Our benchmarking script features automatic device detection to utilize the best available compute backend—CUDA, MPS, or CPU—depending on the platform. Latency is measured using high-resolution timers with `time.perf_counter()` and synchronized using `torch.cuda.synchronize()` where applicable to ensure accurate measurement. The main performance metrics include per-sample latency, inference throughput in FPS, energy consumption per sample, and the serialized model size. All the models are evaluated using pretrained weights on the same 20% test split of the ASL sign dataset. The input resolution is fixed at 224×224 pixels, and inference is run with a batch size of 64 across all platforms to ensure consistency. Additionally, we compute the number of FLOPs for each model using the `ptflops` profiling library to quantify computational complexity.

This benchmarking setup provides a standardized, fair, and reproducible evaluation framework, allowing us to directly compare the performance of `SignNet-Nano` against SOTA models across heterogeneous deployment environments. The results demonstrate that `SignNet-Nano` is highly suitable for real-time ASL recognition on constrained edge hardware while maintaining competitive performance.

V. Performance Evaluation

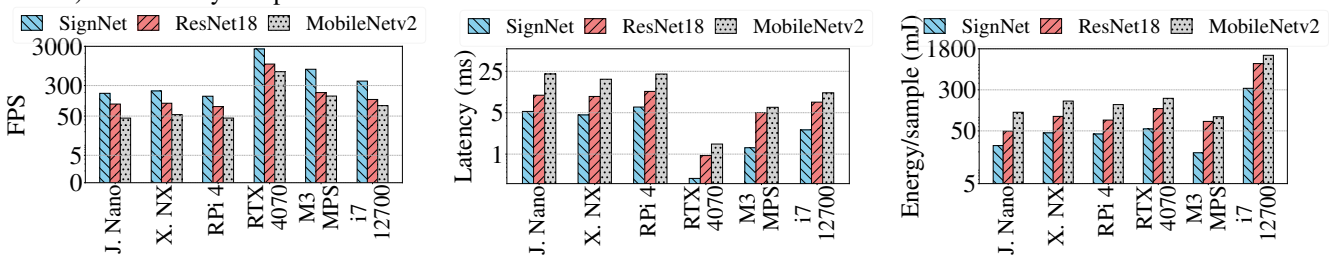
A. Classification Performance

We evaluate the classification performance of `SignNet-Nano` in comparison with two widely used



(a) Number of model parameters (b) Computational burden in terms of FLOPs (c) Memory footprint

Fig. 5: Performance comparison of `SignNet-Nano`, ResNet18 and MobileNetV2 in terms of parameters, computation burden (FLOPs) and memory footprint.



(a) Throughput comparison in terms of FPS (b) Latency analysis (c) Energy consumption analysis

Fig. 6: Throughput, latency and energy consumption analysis across heterogenous edge and high-performance machines.

TABLE I: Comparison of classification performance across different models

Model	Accuracy (%)	Precision	Recall	F1 Score
SignNet-Nano	99.07	0.9908	0.9907	0.9907
ResNet18	100.00	1.0000	1.0000	1.0000
MobileNetV2	99.99	0.9999	0.9999	0.9999

TABLE II: Ablation study showing the impact of architectural components on accuracy, FLOPs, and parameter count.

Variant	Accuracy (%)	FLOPs (M)	Params
SignNet-Nano (Full)	**99.07**	**106.2**	**17.4K**
w/o SE Module	98.62	106.2	16.8K
w/o GAP (FC Layer Added)	97.91	128.5	38.9K
w/o DS Convolutions	97.23	402.8	78.2K

lightweight edge models— ResNet18 and MobileNetV2, employing the ASL dataset on the basis of standard metrics: accuracy, precision, recall, and F1 score. `SignNet-Nano` demonstrates competitive accuracy and F1-score with values of 99.07% in accuracy and 0.9907 in F1-score. Table I depicts that the `SignNet-Nano` achieves performance within 1% of the more resource-intensive models, indicating its capability to effectively preserve prediction performance despite its ultra-lightweight design.

B. Ablation Study

To assess the impact of each core component in `SignNet-Nano`, we conduct an ablation study by selectively removing architectural elements and measuring changes in accuracy, FLOPs, and parameter count, as shown in Table II. Removing the squeeze-and-excitation (SE) module results in a 0.45% drop in accuracy, confirming the benefit of channel-wise attention with minimal overhead. Replacing GAP with a fully connected layer increases parameter count by over 2× and leads to a performance drop, likely due to overfitting caused by the denser feature representation. Finally, replacing DS convolutions with standard convolutions causes a nearly 4× increase in FLOPs and a 4.5× increase in parameters, without any substantial gain in accuracy. These findings validate the architectural efficiency of SignNet-Nano and highlight the necessity of its design choices for lightweight deployment.

C. Real-World Deployment Performance

1) Computational Burden and Memory Footprint: The deployment efficiency of `SignNet-Nano` across a diverse set of hardware platforms in comparison with the other two SOTA models is depicted in Fig. 5. In terms of computational burden, `SignNet-Nano` incurs only 106.2M FLOPs and comprises 17.4K parameters, which corresponds to an 83.4% and 99.5% reduction, respectively, in comparison to MobileNetV2. When benchmarked against ResNet18, `SignNet-Nano` demonstrates a 97.1% reduction in FLOPs and a 99.85% reduction in parameter count, underscoring the architectural efficiency for resource-constrained inference scenarios. Moreover, as depicted in Fig. 5c, `SignNet-Nano` offers the minimal memory footprint of only around 0.1 MB, making it 110× lower than MobileNetV2 (8.8 MB) and 535× smaller than ResNet18 (42.8 MB).

2) Throughput (FPS): In terms of inference throughput, `SignNet-Nano` consistently delivers the highest frame rates across all platforms as presented in Fig. 6a. The advantage becomes more pronounced on high-end devices like RTX 4070— it achieves a peak throughput of 2566.9 FPS, which is a 142.5% improvement over ResNet18 (1058.2 FPS)

and 280% over MobileNetV2 (675.9 FPS). Similar performance advantages are observed on other high-performance systems such as Apple M3 and Intel Core i7, where SignNet-Nano maintains frame rates well above real-time requirements. On edge platforms, including Jetson Nano, Jetson Xavier NX, and Raspberry Pi 4, SignNet-Nano continues to outperform both ResNet18 and MobileNetV2 by a significant margin. This ability to maintain high FPS across heterogeneous hardware makes SignNet-Nano particularly effective for latency-sensitive and real-time sign language detection for inclusive communication.

3) Latency: As depicted in Fig. 6b, SignNet-Nano demonstrates minimal and stable inference latency across all platforms, with the lowest response time on RTX 4070, where it achieves the latency of 0.389 ms, representing a 58.8% and 73.7% reduction compared to ResNet18 (0.945 ms) and MobileNetV2 (1.479 ms), respectively. Across other high-performance systems such as Apple M3 (4.58 ms) and Intel Core i7 (4.61 ms), latency remains under 6 ms for SignNet-Nano, maintaining a consistent advantage over other competing models. On edge platforms including Jetson Nano, Jetson Xavier NX and Raspberry Pi 4, the latency of SignNet-Nano remains stable within the 4–6 ms range. In contrast, baseline models show a sharper increase in latency as compute resources decrease. However, SignNet-Nano achieves the lowest latency across all the devices, highlighting the architectural efficiency and robustness of the proposed model in sustaining real-time responsiveness, regardless of hardware constraints.

4) Energy Efficiency: As shown in Fig. 6c, our proposed model exhibits the lowest energy consumption across all evaluated platforms, making it highly suitable for energy-constrained edge deployments. The most efficient result is achieved on the Apple M3, where SignNet-Nano consumes only 0.0192 J per inference, marking a 74.4% and 79.3% reduction compared to ResNet18 (0.0756 J) and MobileNetV2 (0.0926 J), respectively. This energy advantage persists across both high-end systems and edge devices. For instance, on Jetson Xavier NX, SignNet-Nano consumes 0.0458 J, which is 51.3% and 75.1% lower than ResNet18 (0.0941 J) and MobileNetV2 (0.1838 J), respectively. Similar trends are observed across Jetson Nano, Raspberry Pi 4, and RTX 4070 platforms. These results validate the scalability and efficiency of our model, ensuring energy-aware real-time inference irrespective of the computational budget or platform constraints.

VI. Conclusion and Future Work

In this paper, we introduced SignNet-Nano, an ultra-lightweight CNN architecture specifically designed for real-time ASL recognition on resource-constrained edge devices. By leveraging DS convolutions, SE attention, and GAP, SignNet-Nano achieves a compact design with under 20K parameters and a model size less than 0.1 MB, while maintaining accuracy within 1% of significantly larger state-of-the-art models. We evaluated SignNet-Nano across six heterogeneous platforms, including both high-performance systems and embedded edge devices, and demonstrated its consistent superiority in terms of latency, throughput, and energy efficiency over the existing light-weight SOTA models.

Looking forward, we aim to extend the core design principles of SignNet-Nano to a broader range of real-time computer vision tasks beyond ASL recognition, including vision-based human-computer interaction. Additionally, we plan to explore model generalization across diverse datasets and environmental conditions, ensuring robustness against variations in lighting, background, and camera hardware.

References

[1] M. Sen and R. Rajkumar, "Fostering inclusive communication: A tool integrating machine translation, nlp, and audio-to-sign-language conversion for the deaf," in *2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*. IEEE, 2024, pp. 1–6.

[2] B. Kaur, A. Chaudhary, S. Bano, Yashmita, S. Reddy, and R. Anand, "Fostering inclusivity through effective communication: Real-time sign language to speech conversion system for the deaf and hard-of-hearing community," *Multimedia Tools and Applications*, vol. 83, no. 15, pp. 45 859–45 880, 2024.

[3] K. F. Haque, F. Meneghello, M. E. Karim, and F. Restuccia, "Sawec: Sensing-assisted wireless edge computing," *arXiv preprint arXiv:2402.10021*, 2024.

[4] K. F. Haque, F. Meneghello, and F. Restuccia, "Integrated sensing and communication for efficient edge computing," in *2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2024, pp. 611–614.

[5] M. A. Rupp, "Is it getting hot in here? the effects of vr headset microclimate temperature on perceived thermal discomfort, vr sickness, and skin temperature," *Applied Ergonomics*, vol. 114, p. 104128, 2024.

[6] K. Selvan, M. Mina, H. Abdelmeguid, M. Gulsha, A. Vincent, and A. Sarhan, "Virtual reality headsets for perimetry testing: a systematic review," *Eye*, vol. 38, no. 6, pp. 1041–1064, 2024.

[7] J. Soni, "Apple vision pro vs. meta quest 3: The next generation of vr," Jan 2024. [Online]. Available: https://www.dexerto.com/tech/apple-vision-pro-vs-meta-quest-3-2168054/

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

[10] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[11] A. Kasapbaşi, A. E. A. Elbushra, A.-H. Omar, and A. Yilmaz, "Deepaslr: A cnn based human computer interface for american sign language recognition for hearing-impaired individuals," *Computer methods and programs in biomedicine update*, vol. 2, p. 100048, 2022.

[12] S. Sharma and K. Kumar, "Asl-3dcnn: American sign language recognition technique using 3-d convolutional neural networks," *Multimedia Tools and Applications*, vol. 80, no. 17, pp. 26 319–26 331, 2021.

[13] G. Z. De Castro, R. R. Guerra, and F. G. Guimarães, "Automatic translation of sign language with multi-stream 3d cnn and generation of artificial depth maps," *Expert Systems with Applications*, vol. 215, p. 119394, 2023.

[14] V. Sharma, A. Sharma, and S. Saini, "Real-time attention-based embedded lstm for dynamic sign language recognition on edge devices," *Journal of Real-Time Image Processing*, vol. 21, no. 2, p. 53, 2024.

[15] V.-E. Baciu, J. Stiens, and B. da Silva, "Mlino bench: A comprehensive benchmarking tool for evaluating ml models on edge devices," *Journal of Systems Architecture*, vol. 155, p. 103262, 2024.

[16] A. Nagaraj, "Asl alphabet," https://www.kaggle.com/datasets/grassknoted/asl-alphabet, 2018, dOI: 10.34740/KAGGLE/DSV/29550.