Computer Vision for ASL Recognition and Translation

Abhishek Nagraj Bijoor
University of Texas
Arlington
axn8253@mavs.uta.edu

Suhas Kowligi
University of Texas
Arlington
sxk7070@mavs.uta.edu

Abstract

We propose a deep learning model for American Sign Language (ASL) recognition and translation. This study will recognize ASL fingerspelling, exploring effective solutions for ASL users. We explore the development of an efficient ASL classification system by leveraging deep learning techniques, particularly Convolutional Neural Networks (CNNs). The system incorporates feature extraction and augmentation techniques to handle variations in lighting, angles, and positions. The models are trained and evaluated using performance metrics such as accuracy, precision, recall and F1-score. Experimental results demonstrate the proposed system's ability to achieve high classification accuracy while avoiding over-fitting. This study emphasizes the challenges posed by occlusion, diverse backgrounds, and signer variations, offering solutions through data augmentation, and fine-tuning of models. The proposed ASL classification framework has the potential to be integrated into real-world applications, including mobile applications, wearable devices, and assistive technologies, thereby enhancing accessibility.

1. Introduction

Communication barriers between deaf or hard-of-hearing individuals who use American Sign Language (ASL) and those who do not understand ASL remain a significant challenge in many social and professional environments. Our goal is to address this challenge by utilizing computer vision and machine learning techniques to create a model capable of recognizing and translating ASL fingerspelling. The proposed model is designed to offer efficient performance and flexibility in deployment across diverse platforms. This model will enhance communication in a range of environments, including educational settings and public services. Key use cases include assisting ASL users in daily interactions by translating signs into spoken language, providing learners with feedback on their ASL gestures, and supporting communication in face-to-face or virtual meetings.

2. Related Work

The recognition and classification of American Sign Language (ASL) gestures have been widely explored in the field of human-computer interaction and computer vision. Early approaches relied on handcrafted features and traditional machine learning algorithms.

For instance, parallel **Hidden Markov Models (HMMs)** [1] have been effectively utilized to capture the sequential and simultaneous aspects of ASL recognition. [Christian Vogler et al., 2000] proposed a framework for continuous ASL recognition based on linguistic principles, particularly the phonology of ASL. In this approach, each HMM represents a single phoneme, allowing for scalability given the finite number of phonemes compared to the vast number of ASL signs that can be composed from them. This phoneme-based framework demonstrated potential for broader applications, including scaling to larger and more complex ASL datasets.

Advancements in deep learning have significantly contributed to the development of ASL recognition systems. A comprehensive study by [Alsharif B et al., 2023] explored the application of five distinct deep learning models—AlexNet, ConvNeXt, EfficientNet, ResNet-50, and VisionTransformer—to recognize ASL alphabet hand gestures. The models were trained and tested on a dataset of over 87,000 images, yielding remarkable results. ResNet-50 achieved the highest accuracy at 99.98%, followed by EfficientNet at 99.95%, while VisionTransformer produced the lowest accuracy at 88.59%. This study underscores the potential of

modern architectures in achieving high accuracy rates for ASL gesture recognition and highlights ResNet-50′s superior performance in this domain [2].

Recurrent Neural Networks (RNNs), have proven their significance in recognizing the spatial and temporal features of ASL. The combination of CNNs, such as Inception, for spatial feature extraction, along with RNNs for temporal sequence modeling, has been widely explored. Previous work by [Kshitij Bantupalli Ying Xie, 2024] demonstrated a vision-based sign language translation model that integrates these techniques to bridge the communication gap between signers and non-signers [3].

[Sruthi C. J and Lijiya A., 2019] proposed a CNN-based approach for static Indian Sign Language (ISL) alphabet recognition, where the binary silhouette of the hand region was used as input. This approach demonstrated high accuracy, with the model achieving 98.64%, outperforming traditional methods like Support Vector Machines (SVMs) and Hidden Markov Models (HMMs). The use of CNNs enabled the model to capture the spatial features of hand gestures, offering a unique solution for signer-independent ISL recognition [4].

These studies showcase the evolution of ASL recognition methods, moving from traditional machine learning approaches to deep learning techniques, while also revealing the persistent challenges in real-world datasets. The deep learning techniques used in these studies guided us towards our work in ASL recognition.

3. Problem Statement

Communication is a fundamental human need, yet individuals with hearing impairments often face significant challenges in interacting seamlessly with those who do not understand sign language. American Sign Language (ASL) serves as a primary means of communication for many within the Deaf and Hard of Hearing community. However, the lack of widespread ASL knowledge among the general population creates a communication gap, limiting accessibility and inclusivity.

The problem of American Sign Language Recognition (ASLR) lies in accurately interpreting ASL gestures, including both static hand signs and dynamic motion sequences, into readable or spoken language. Traditional methods are often hindered by limited scalability, high computational requirements, and difficulty in adapting to signer variability, noise and occlusions.

This study will utilize publicly available dataset for ASL gesture recognition: **ASL Alphabet Dataset** [5]: The data set is a collection of images of alphabets from the American Sign Language, separated in 29 folders which represent the various classes. The training data set contains 87,000 images which are 200x200 pixels. There are 29 classes, of which 26 are for the letters A-Z and 3 classes for SPACE, DELETE and NOTHING. These 3 classes are very helpful in real-time applications, and classification. The test data set contains a mere 29 images, to encourage the use of real-world test images.

The ASLR system aims to deliver high performance in recognizing American Sign Language gestures. A primary objective is to attain recognition accuracy exceeding 90% for static gestures, particularly the individual alphabet signs. This high level of accuracy is pivotal for reliable communication and interpretation of ASL.

Beyond raw accuracy, the system is expected to maintain good performance across a variety of real-world conditions. This includes consistent recognition capabilities when faced with diverse signers, varying lighting conditions, and different background environments. Such versatility is essential for the system's practical application in diverse settings.

To assess the system's effectiveness, a robust evaluation approach will be employed. The primary metric will be **overall accuracy**, calculated as the percentage of correctly classified gestures. We also analyze the **confusion matrix** which allows evaluation for each class in a multi-class problem such as this, helping identify which classes are being predicted accurately and where the model struggles. This provides a straightforward measure of the system's general performance.

However, to get deeper insights into the system's capabilities, especially in handling potential dataset imbalances, additional metrics will be examined. These include **precision, recall and F1-score**. Another aspect of the evaluation process will be testing the system's generalizability. This involves assessing performance on unseen data, including gestures from signers not included in the training set and scenarios that vary from the training distribution.

4. Problem Solution

4.1. Approach

To address the problem of American Sign Language (ASL) recognition, we developed a cus-

tom Convolutional Neural Network (CNN) model, `Custom_ASL_CNN`. This solution leverages deep learning to extract spatial hierarchies and patterns from image data for accurate ASL sign classification. The network architecture includes convolutional layers for feature extraction, batch normalization to stabilize and accelerate training, pooling layers for dimensionality reduction, and fully connected layers for final classification. Data augmentation on the input images was also utilized to enhance generalization.

4.2. System Architecture

The architecture for the ASL recognition problem consists of several key components that work to process input data, extract meaningful features, and classify ASL gestures. The architecture is designed to ensure high accuracy and efficiency in recognizing 29 distinct ASL signs.

The proposed system for ASL gesture recognition begins with an input layer that accepts images of ASL gestures. These images are resized to a fixed resolution, **224x224x3**, and normalized to ensure consistency across the dataset.

A comprehensive preprocessing pipeline follows, implemented using torchvision.transforms. This pipeline includes data augmentation techniques such as random horizontal flipping, rotation, affine transformations, and color jittering. These transformations increase dataset variability, enhancing the model's ability to generalize. The preprocessing also involves normalization, adjusting each RGB channel to have a mean of 0.5 and a standard deviation of 0.5. This process ensures the model's invariance to input distortions, as illustrated below.

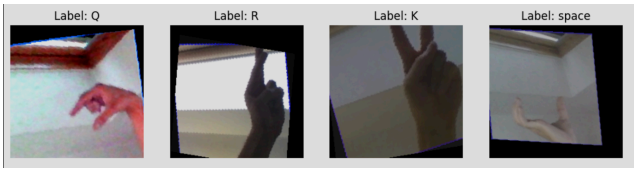

Figure 1. Data Augmentation Results

At the core is the `Custom_ASL_CNN` model, specifically designed for ASL recognition. This model strikes a balance between depth and computational efficiency. It involves five convolutional layers that extract hierarchical spatial features, progressing from low-level textures to high-level patterns. Each convolutional layer is followed by **batch normalization** [6], which stabilizes training and improves convergence. Max-pooling layers are placed in between layers to reduce spatial dimensions while retaining important features. The network concludes with three fully connected layers that map the extracted features to class probabilities for 29 ASL signs. **Dropout layers** [7] are integrated after the fully connected layers to address the challenge of overfitting. During inference, the model's output logits are passed through a softmax operation to generate class probabilities.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 55, 55]	23,296
BatchNorm2d-2	[-1, 64, 55, 55]	128
MaxPool2d-3	[-1, 64, 27, 27]	0
Conv2d-4	[-1, 192, 27, 27]	307,392
BatchNorm2d-5	[-1, 192, 27, 27]	384
MaxPool2d-6	[-1, 192, 13, 13]	0
Conv2d-7	[-1, 384, 13, 13]	663,936
BatchNorm2d-8	[-1, 384, 13, 13]	768
Conv2d-9	[-1, 256, 13, 13]	884,992
BatchNorm2d-10	[-1, 256, 13, 13]	512
Conv2d-11	[-1, 256, 13, 13]	590,080
BatchNorm2d-12	[-1, 256, 13, 13]	512
MaxPool2d-13	[-1, 256, 6, 6]	0
Linear-14	[-1, 4096]	37,752,832
Dropout-15	[-1, 4096]	0
Linear-16	[-1, 4096]	16,781,312
Dropout-17	[-1, 4096]	0
Linear-18	[-1, 29]	118,813

Total params: 57,124,957
Trainable params: 57,124,957
Non-trainable params: 0

Figure 2. `Custom_ASL_CNN` summary

4.3. Experiments

4.3.1. Training Custom_ASL_CNN

The training process for the Custom_ASL_CNN employs several optimization techniques to ensure efficient and effective learning. The dataset is split into training (80%) and validation (20%) subsets, allowing for robust model evaluation during training. The best model weights are saved after each epoch based on validation performance, ensuring that the final model represents the best-performing iteration.

The **Adam optimizer** [8] is utilized with an initial learning rate of 10^{-4}, chosen for its adaptive learning rate capabilities. A `ReduceLROnPlateau` scheduler is implemented to reduce the learning rate when the validation loss plateaus, aiding in more refined parameter updates as training progresses.

The loss function chosen for this multi-class classification task is **Cross-Entropy Loss** [9], defined as:

$$H(p, q) = -\sum_{x \in X} p(x) \log(q(x)) \tag{1}$$

Where $H(p, q)$ represents the cross-entropy between the true probability distribution p and the predicted distribution q, and X is the set of possible outcomes.

To prevent over-fitting and unnecessary computation, an early stopping mechanism is employed with a patience of 3 epochs. This is implemented using the following parameters:

```
early_stopping_patience = 3
```

The training process is set to run for a maximum of **10 epochs**. During each epoch, the validation loss is compared to the best validation loss observed so far. If the current validation loss is lower, it becomes the new best validation loss, and the model weights are saved. If the validation loss does not improve, the `no_improve_epochs` counter is incremented. When this counter reaches the `early_stopping_patience` value of 3, training is halted. This approach ensures that the model does not overfit to the training data and stops training when no further improvement is observed, saving time and resources. The combination of adaptive learning rate, early stopping, and best model saving creates a solid framework for accurate ASL gesture recognition while mitigating common training pitfalls such as over-fitting.

4.3.2. Algorithm for early stopping

```
if avg_val_loss < best_val_loss:
    best_val_loss = avg_val_loss
    no_improve_epochs = 0

    # Save the best model
    torch.save(model.state_dict(),
"best_weights.pth")
else:
    no_improve_epochs += 1
    if    no_improve_epochs    >=
early_stopping_patience:
      break
```

4.3.3. Results with Test Dataset

The confusion matrix for the original test set demonstrates strong performance with most predictions aligning perfectly with the actual classes. Misclassifications were minimal, showcasing high accuracy overall.

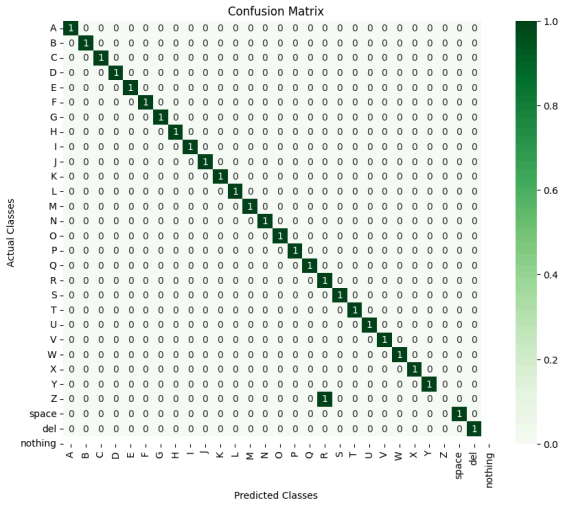

Figure 3. Confusion Matrix with original testset

Test Accuracy: **96.43%** .

The performance across all classes was evaluated using precision, recall and F1-score metrics. Below are the summarized results:

Class	Precision	Recall	F1-Score
0	1.00	1.00	1.00
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	1.00	1.00	1.00
6	1.00	1.00	1.00
7	1.00	1.00	1.00
8	1.00	1.00	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	1.00	1.00
12	1.00	1.00	1.00
13	1.00	1.00	1.00
14	1.00	1.00	1.00
15	1.00	1.00	1.00
16	1.00	1.00	1.00
17	0.50	1.00	0.67
18	1.00	1.00	1.00
19	1.00	1.00	1.00
20	1.00	1.00	1.00
21	1.00	1.00	1.00
22	1.00	1.00	1.00
23	1.00	1.00	1.00
24	1.00	1.00	1.00
25	0.00	0.00	0.00
26	1.00	1.00	1.00
28	1.00	1.00	1.00

Table 1. Performance Metrics

The results demonstrate a significant performance with minimal error on the mere test set of 29 images, highlighting the efficacy of the proposed approach without over-fitting.

4.3.4. Results with Additional Real World Data

Since the test set does not have enough data for verification, we added the following "real-world" images during evaluation:

Figure 4."A"

Figure 5."B"

Figure 6."del"

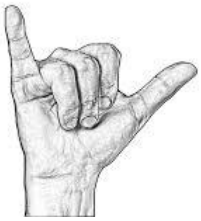

Figure 7."Y"

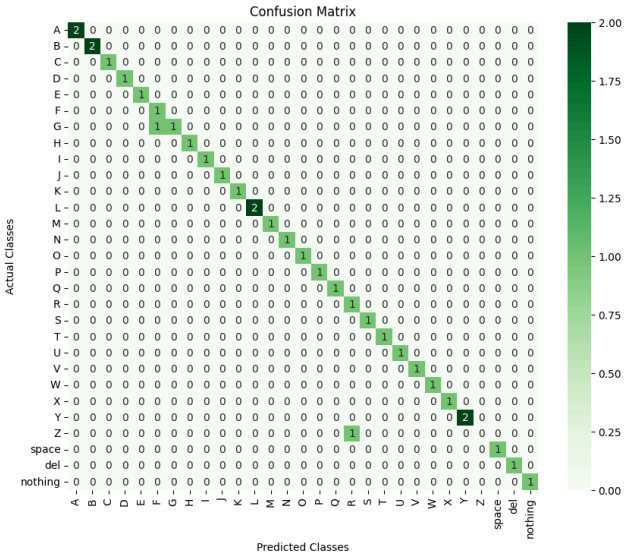

Figure 8. Confusion Matrix with real-world images

Real-world Test Accuracy: **94.12%** .

Class	Precision	Recall	F1-Score
0	1.00	1.00	1.00
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	0.50	1.00	0.67
6	1.00	0.50	0.67
7	1.00	1.00	1.00
8	1.00	1.00	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	1.00	1.00
12	1.00	1.00	1.00
13	1.00	1.00	1.00
14	1.00	1.00	1.00
15	1.00	1.00	1.00
16	1.00	1.00	1.00
17	0.50	1.00	0.67
18	1.00	1.00	1.00
19	1.00	1.00	1.00
20	1.00	1.00	1.00
21	1.00	1.00	1.00
22	1.00	1.00	1.00
23	1.00	1.00	1.00
24	1.00	1.00	1.00
25	0.00	0.00	0.00
26	1.00	1.00	1.00
27	1.00	1.00	1.00
28	1.00	1.00	1.00

Table 2. Real-world Performance Metrics

As we can see, it is evident that the model is performing good with real-world data and shows minimal signs of over-fitting.

4.3.5. Fine-tuning Baseline models

To continue with the experiment, we chose to fine-tune the last layers of AlexNet and ResNet34 for baseline model comparison. To maintain fairness, both the models were tested on the same "real-world" dataset used for Custom_ASL_CNN model. The input size were adjusted as per the model's expectations; i.e. **(227x227x3)** for AlexNet and **(224x224x3)** for ResNet34 respectively. Considering the lack of compute resources, these baseline models were fine-tuned without data augmentations.

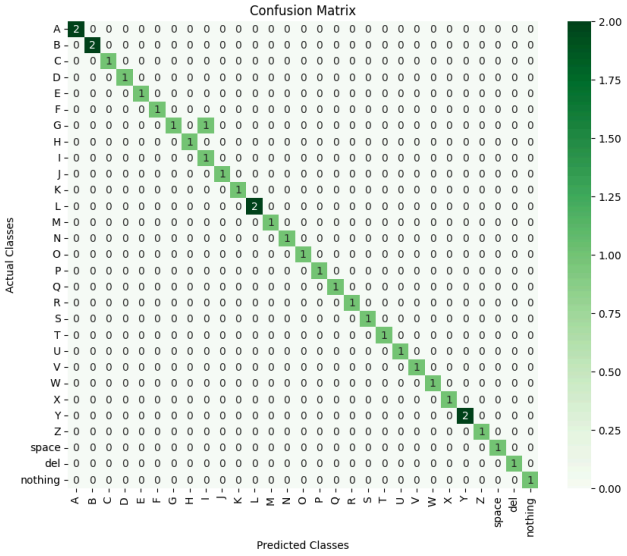

Figure 9. Confusion Matrix for AlexNet

AlexNet Test Accuracy: **97.06%** .

Class	Precision	Recall	F1-Score
0	1.00	1.00	1.00
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	0.50	1.00	0.67
6	1.00	0.50	0.67
7	1.00	1.00	1.00
8	1.00	1.00	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	1.00	1.00
12	1.00	1.00	1.00
13	1.00	1.00	1.00
14	1.00	1.00	1.00
15	1.00	1.00	1.00
16	1.00	1.00	1.00
17	0.50	1.00	0.67
18	1.00	1.00	1.00
19	1.00	1.00	1.00
20	1.00	1.00	1.00
21	1.00	1.00	1.00
22	1.00	1.00	1.00
23	1.00	1.00	1.00
24	1.00	1.00	1.00
25	0.00	0.00	0.00
26	1.00	1.00	1.00
27	1.00	1.00	1.00
28	1.00	1.00	1.00

Table 3. Performance Metrics - AlexNet

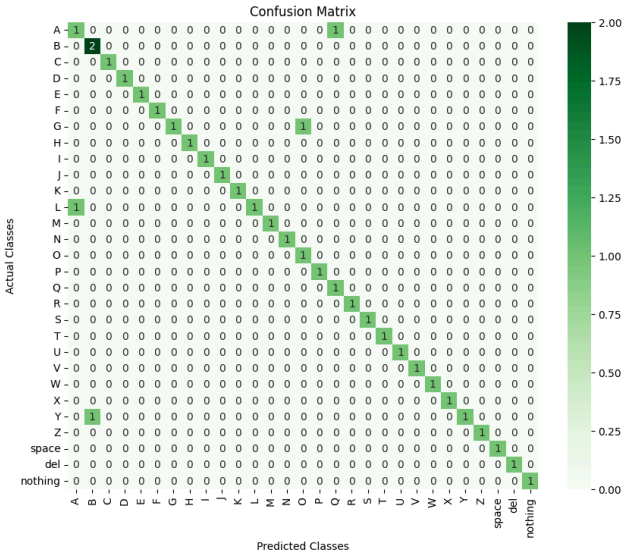

Figure 10. Confusion Matrix for ResNet34

ResNet34 Test Accuracy: **88.24%** .

Class	Precision	Recall	F1-Score
0	1.00	0.50	0.67
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	0.50	1.00	0.67
6	1.00	0.50	0.67
7	1.00	1.00	1.00
8	1.00	1.00	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	0.67	1.00	0.80
12	1.00	1.00	1.00
13	1.00	1.00	1.00
14	1.00	1.00	1.00
15	0.50	1.00	0.67
16	1.00	1.00	1.00
17	1.00	1.00	1.00
18	1.00	1.00	1.00
19	1.00	1.00	1.00
20	1.00	1.00	1.00
21	1.00	1.00	1.00
22	1.00	1.00	1.00
23	1.00	1.00	1.00
24	1.00	0.50	0.67
25	1.00	1.00	1.00
26	1.00	1.00	1.00
27	1.00	1.00	1.00
28	1.00	1.00	1.00

Table 4. Performance Metrics - ResNet34

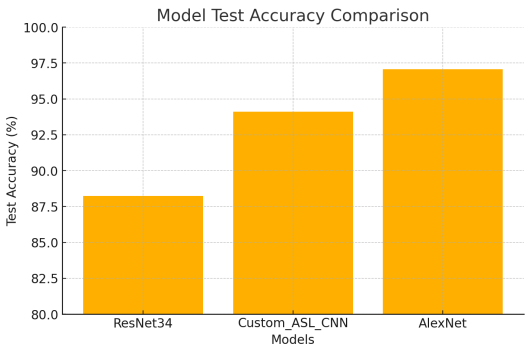

Figure 11. Accuracy of models

5. Conclusion

In this work, we evaluated the performance of several deep learning architectures, including AlexNet, ResNet34, and also trained a customized CNN model, on a large-scale ASL dataset comprising 87,000 images. Our results demonstrate that the custom model outperformed ResNet34, highlighting its effective adaptation to the nuances of ASL gesture recognition. Interestingly, ResNet34 exhibited lower test accuracy, which can be attributed to the absence of data augmentation during training. This highlights the importance of augmentation techniques in enhancing generalization, particularly for deeper architectures like ResNet34.

The superior performance of AlexNet suggests that simpler architectures can excel in specific tasks such as ASL recognition, while the custom model underscores the potential of tailored solutions. Future work could focus on incorporating data augmentation, optimizing preprocessing techniques, and leveraging transfer learning to further improve recognition accuracy. These efforts aim to establish a scalable framework for real-time ASL interpretation systems.

References

[1] Christian Vogler, H. Sun, and Dimitris Metaxas, A framework for motion recognition with applications to Americansign language and gait recognition. pages 33–38, 2000. doi: 10.1109/HUMO.2000.897368.

[2] Bader Alsharif, Ali Altaher, Ahmed Altaher, Mohammad Ilyas, and Easa Alalwany, Deep Learning Technology to Recognize American Sign Language Alphabet, vol. 23. page , 2023. doi: 10.3390/s23187970.

[3] Kshitij Bantupalli and Ying Xie, American Sign Language Recognition using Deep Learning and Computer Vision, vol. 0, no. , pages 4896–4899, 2018, doi: 10.1109/BigData.2018.8622141.

[4] Sruthi C J and A. Lijiya, Signet: A Deep Learning based Indian Sign Language Recognition System, pages 596–600, 2019, doi: 10.1109/ICCSP.2019.8698006.

[5] Akash Nagaraj, ASL Alphabet, Kaggle, 2018. doi: 10.34740/KAGGLE/DSV/29550.

[6] Hanyang Peng, Yue Yu, and Shiqi Yu, Re-Thinking the Effectiveness of Batch Normalization and Beyond, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 1, pages 465–478, 2024, doi: 10.1109/TPAMI.2023.3319005.

[7] Asma ElAdel, Ridha Ejbali, Mourad Zaied, and Chokri Ben Amar, Fast deep neural network based on intelligent dropout and layer skipping, in *2017 International Joint*

Conference on Neural Networks (IJCNN), 2017, pages 897–902. doi: 10.1109/IJCNN.2017.7965947.

[8] Diederik P. Kingma and Jimmy Ba, Adam: A Method for Stochastic Optimization, 2017, [Online]. Available: https://arxiv.org/abs/1412.6980

[9] Yaoshiang Ho and Samuel Wookey, The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling, *IEEE Access*, vol. 8, no. , pages 4806–4813, 2020, doi: 10.1109/ACCESS.2019.2962617.