Masked Batch Normalization to Improve Tracking-Based Sign Language Recognition Using Graph Convolutional Networks

Natsuki Takayama[1] and Gibran Benitez-Garcia[1] and Hiroki Takahashi[1,2]
[1] Graduate School of Informatics and Engineering, the University of Electro-Communications, Tokyo, Japan
[2] Artificial Intelligence eXploration Research Center, Tokyo, Japan

2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021) | 978-1-6654-3176-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/FG52635.2021.9667007

***Abstract*— Sign language recognition is a fundamental technique to improve communication between native signers and speakers. Current state-of-the-art sign language recognition methods often apply deep neural network models to learn an optimized projection between sign language videos and sentences in an end-to-end manner. Generally, minibatch training using sequential data requires the addition of padding to equalize the varying lengths of sequences. However, this training strategy induces performance degradation if batch normalization is used in the models because batch normalization assumes the validity of all inputs. In this study, we propose masked batch normalization, which normalizes input features while masking dummy signals. We apply masked batch normalization to tracking-based sign language recognition models using graph convolutional networks. The performance of the proposed method is evaluated in isolated sign language word recognition and continuous sign language words recognition settings. To evaluate the proposed method, we use two types of sign language video datasets, WLASL including 2000 types of isolated words, and a JSL dataset including 275 types of videos of isolated words and 113 types of videos showing sentences. The evaluation results show that the proposed method improves the tracking-based sign language recognition models in both cases.**

I. Introduction

Sign language recognition, which estimates words represented by sign motions, is an important technique for improving communication between native signers and speakers. As a result of continuous research efforts over three decades on vision-based sign language recognition, continuous sign language words recognition methods based on deep neural networks (DNNs) have been proposed in recent years [1], [2], [3], [4].

However, sign language recognition commonly encounters issues in handling variable-length sequences. Although models can process variable-length sequences, they require minibatch sequences of a fixed length in training. Moreover, several types of the sequence to sequence (seq2seq) models, such as Transformer, assume fixed-length sequences as inputs. Padding techniques appending dummy signals to equalize the length of sequences are commonly used to handle this situation. However, if a model includes normalization layers using statistical values such as batch normalization (BN) layers [5], the addition of dummy signals will degrade its recognition performance.

BN is a standard DNN technique used in a wide variety of tasks, including sign language recognition. However, BN assumes the validity of all inputs, and the dummy signals associated with padding are not considered. The effect of such dummy signals on BN and the necessity of masking them have been discussed in software communities[1], but the effects of such methods on model performance have not as yet been evaluated.

Based on the above background, we propose masked batch normalization (MBN) to extend BN by masking the dummy signals. Training convolutional neural network (CNN)-based models using video frames requires considerable computational resources, and in some cases, it is sometimes infeasible owing to the complex pre-training processes required. Therefore, we apply MBN to tracking-based sign language recognition models using graph convolution networks (GCN). GCN is a generalization of CNN to a graph structure, and it has attracted attention in tracking-based action recognition owing to its high recognition performance. Most types of GCN include BN layer in those blocks, and thus, they are suitable for applying the proposed method. Moreover, training the tracking-based models is notably lightweight compared to that of the CNN-based models because they use tracking points instead of video frames. As a result, they can be trained from scratch with consumer graphic processing units.

Sign language recognition includes isolated sign language word recognition and continuous sign language words recognition tasks. We employ spatial-temporal graph convolution network (STGCN) [6] and sign language graph convolution network (SLGCN) [7] as base models for the isolated sign language word recognition. STGCN is the first application of GCN to tracking-based action recognition, and it has been the basis for subsequent methods. SLGCN is an extension of STGCN, including decoupling GCN (DGCN) with Drop Graph [8] and STC Attention modules [9], and it achieves the state-of-the-art performance for isolated sign language word recognition in recent years. Furthermore, we extend them for the continuous sign language words recognition by combining them with Transformer [10] similar to [4]. We apply MBN to these four types of GCN-based models and investigate performance improvement.

The proposed method was evaluated using WLASL [11], which includes 2000 types of isolated sign language word, and a Japanese sign language (JSL) dataset [4] including 275 and 113 types of videos showing isolated words and

This work was supported by SoftBank Corp.

[1] https://yangkky.github.io/2020/03/16/masked-batch-norm.html
https://discuss.pytorch.org/t/masked-batch-normalization/57813

sentences. Our experimental results indicate that MBN can improve on GCN-based models by a significant margin.

In the following sections, we refer to "isolated sign language word" and "continuous sign language words" as "isolated word" and "continuous words", respectively, for simplicity.

II. Related work

Methods combining framewise feature extraction and temporal recognition are among the standard approaches in sign language recognition. Early methods employed statistical temporal recognition, such as Hidden Markov Models, and combined them with handcrafted features [12], [13], [14]. In recent years, DNNs have replaced these technical elements.

At present, the most standard DNN-based feature extraction methods apply CNN directly to video frames [1], [2], [3]. This approach often employs pre-trained CNN blocks [15], [16], [17] as feature extractors. Tracking-based approaches have also been developed utilizing tracking points extracted from videos, and both CNN and GCN have been employed in these methods [4], [18], [11], [7]. Normalization layers such as BN are often applied in addition to these convolution layers, but as yet existing methods have not addressed their impacts on performance in combination with dummy signals.

Recurrent neural networks with attention mechanisms [19] and Transformer [10] are commonly employed to perform temporal recognition [1], [3], [4], [18], [20]. The Transformer model includes layer normalization [21] in the blocks. Although layer normalization is expected to be affected by the presence of dummy signals similarly to BN, this influence varies depending on the specific implementation[2]. Therefore, we focus on feature extraction using BN in this study.

III. Masked Batch Normalization

BN is a standard DNN technique which normalizes inputs using statistics of minibatch data. BN is commonly applied to stabilize the training processes of DNN models and improve their performance.

Let x_{ncf} and $\hat{x}_{ncf}$ be features before and after normalization, respectively. Then, we denote by $n \in [1, N]$ and $c \in [1, C]$ indices along batch and channel axes, respectively, while $f \in \boldsymbol{F}$ denotes an index of a feature map. The two-dimensional feature map $f = (t, j); t \in [1, T], j \in [1, J], \boldsymbol{F} = [1, T] \times [1, J]$ is used in the GCN layers. t and j are indices of the frames and joints, respectively. The one-dimensional feature map $f = t, \boldsymbol{F} = [1, T]$ is used in other layers.

Using this notation, the normalization is defined as follows.

$$\hat{x}_{ncf} = \gamma_c \frac{x_{ncf} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} + \beta_c, \tag{1}$$

[2]In our observations, some DNN frameworks, such as Tensorflow and PyTorch, implement layer normalization as positional normalization with their default settings, which does not propagate the effect of dummy signals to other positions.

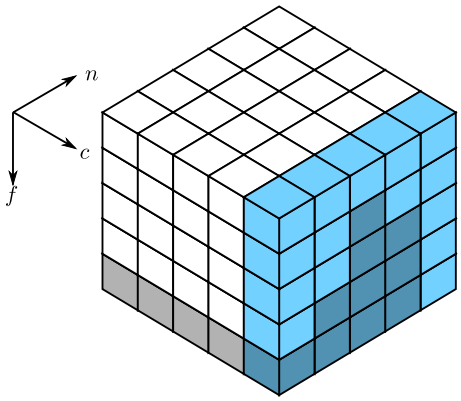


Fig. 1: Batch Normalization, including dummy signals.

where γ_c and β_c are the trainable parameters and ϵ is a fixed value to stabilize the calculation; we use $\epsilon = 1.0 \times 10^{-5}$. μ_c and σ_c denote batch mean and variance, respectively. The batch-mean and batch-variance are calculated for each channel and are defined as follows.

$$\mu_c = \frac{1}{N|\boldsymbol{F}|} \sum_{n,f} x_{ncf}, \tag{2}$$

$$\sigma_c^2 = \frac{1}{N|\boldsymbol{F}|} \sum_{n,f} (x_{ncf} - \mu_c)^2. \tag{3}$$

Fig. 1 shows normalization for channel c. Each voxel indicates an element of a batch x_{ncf}. The elements drawn by light blue voxels are used to calculate μ_c and σ_c^2, and these statistics normalize the elements. The gray voxels indicate dummy signals generated by padding. As shown in Fig. 1, the standard BN includes these dummy signals in calculating statistics.

In contrast, MBN utilizes a mask $m_{nf} \in \{0, 1\}$ to calculate these statistics. $m_{nf} = 1$ and $m_{nf} = 0$ indicate that x_{ncf} is a valid feature or a dummy signal, respectively. In MBN, the batch-mean and batch-variance are calculated as follows.

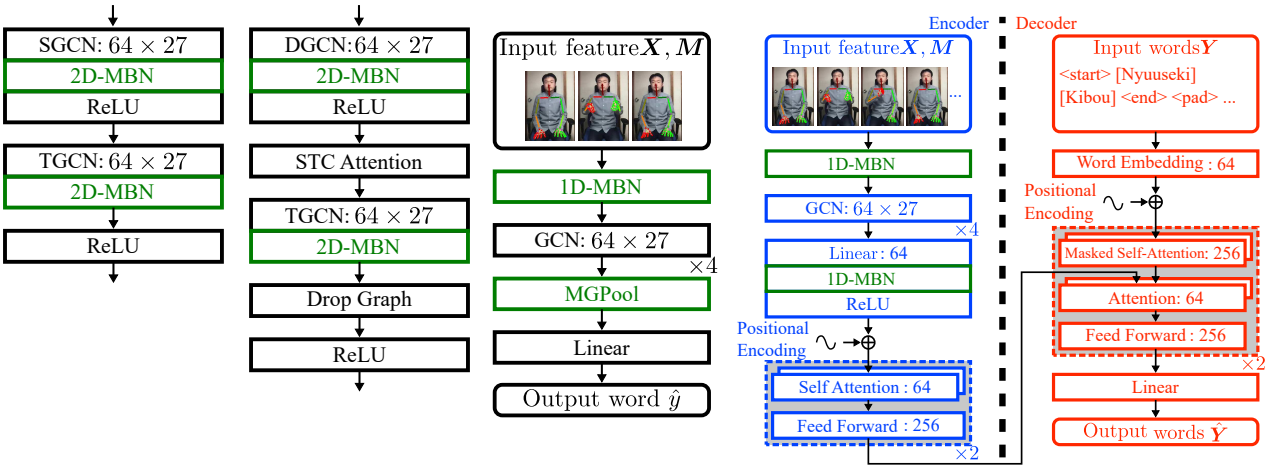
$$\mu_c = \frac{1}{\sum_{n,f} m_{nf}} \sum_{n,f} x_{ncf} m_{nf}, \tag{4}$$

$$\sigma_c^2 = \frac{1}{\sum_{n,f} m_{nf}} \sum_{n,f} (x_{ncf} m_{nf} - \mu_c)^2. \tag{5}$$

The calculation performed by MBN is simple, and it can be implemented by adding the specified mask generation and multiplication operation to the standard BN.

IV. GCN-based sign language recognition

This section describes sign language recognition models in this research. Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_T\}, where\ \boldsymbol{x}_t \in \mathcal{R}^{54}$ and $\boldsymbol{Y} = \{y_1, \ldots, y_s, \ldots, y_S\}, where\ y_s \in \{<\text{start}>, <\text{end}>, <\text{pad}>, \boldsymbol{L}\}$ be input feature and word sequences, respectively. $\boldsymbol{x}_t$ indicates the coordinates of the tracking points in the tth frame. We employ OpenPose [22] for human body-part tracking and use 27 tracking points representing a person's nose, eyes, shoulders, arms, and hands



(a) STGCN block. (b) SLGCN block. (c) Isolated word recognition model. (d) Continuous words recognition model.

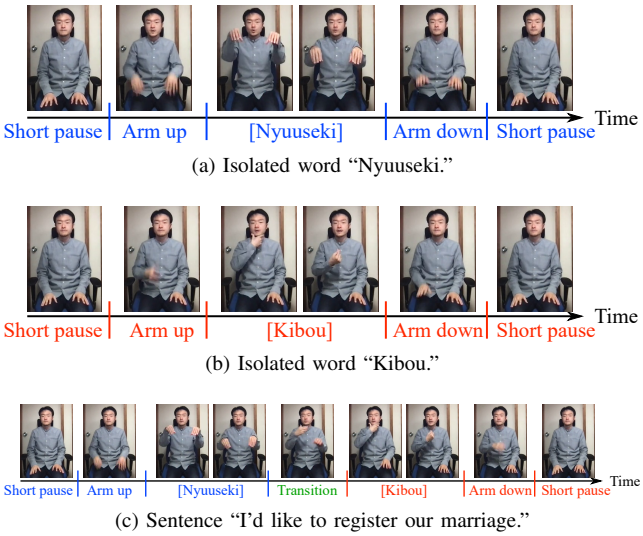
Fig. 2: Illustration of recognition models. MBN and MGPool modules are colored in green. The encoder and decoder of the continuous words recognition model are colored in blue and red, respectively. Rectangles and rounded rectangles indicate layers and data, respectively. The vocabulary determines the dimension of the final Linear layer in (c) and (d).

as input. Hands include metacarpophalangeal joints of index, middle, ring, and pinky fingers, and all fingertips. $\boldsymbol{L}$ denotes a set of words to be recognized. <start > and <end > are keywords indicating the start and end of a word sequence, respectively. <pad > is a keyword for padding. Isolated and continuous words recognition models optimize projection $\boldsymbol{X} \rightarrow L$ and $\boldsymbol{X} \rightarrow \boldsymbol{Y}$ through learning, respectively.

Fig. 2 shows the model structure. Fig. 2 (a) and (b) illustrate STGCN [6] and SLGCN [7] modules, respectively. The residual connections are omitted in Fig. 2 (a) and (b) for simplicity. We employ a spatiotemporal graph connecting tracking points in each frame with spatial edges. The spatial edges follow the graph definition in [7]. The spatial graph convolution (SGCN) and DGCN layers apply their GCN operation according to the spatial edges. Similarly, the spatiotemporal graph connects the same tracking points between frames with the temporal edges. The temporal graph convolution (TGCN) applies temporal CNN operation according to the temporal edges. We replace the BN layers of these modules with MBN layers in this research. We note that the DGCN layer of Fig. 2 (b) includes one MBN layer.

Fig. 2 (c) shows the isolated word recognition model. $\boldsymbol{M}$ indicates the mask for MBN. First, the model normalizes the coordinates using the temporal MBN. Next, four cascaded GCN layers apply graph convolution to the intermediate features. After that, a masked global average pooling (MGPool) is applied. MGPool summarizes sequential features into a 64-dimensional vector while masking padding signals. The computation of MGPool is the same as Eq.(4). Finally, a linear transformation layer is applied to obtain responses to each word.

Fig. 2 (d) shows the continuous words recognition model. A linear transformation layer and temporal MBN are applied after the GCN layers to compress the intermediate features.



(a) Isolated word "Nyuuseki."

(b) Isolated word "Kibou."

(c) Sentence "I'd like to register our marriage."

Fig. 3: Examples of JSL videos.

TABLE I: Summy of the dataset.

Subset types	Training		Test	
# of signers	35		2	
# of isolated words	22640	(275)	3862	(210)
# of sentences	7466	(120)	1372	(105)

Subsequent processes are the same as those of the conventional method [4].

V. Evaluation

A. Dataset

We used two kinds of sign language video datasets to evaluate the proposed method. WLASL is a large-scale American sign language (ASL) words video dataset for isolated word

TABLE II: Recognition performance.

Models	WLASL100		WLASL300		WLASL1000		WLASL2000		JSL	
	BN	MBN	BN	MBN	BN	MBN	BN	MBN	BN	MBN
STGCN	51.55	48.84 $^{\uparrow 2.71}$	63.92	61.15 $^{\uparrow 2.77}$	71.35	69.96 $^{\uparrow 1.39}$	79.81	78.68$^{\uparrow 1.13}$	N/A	N/A
SLGCN	47.29	40.31 $^{\uparrow 6.98}$	63.10	51.50 $^{\uparrow 11.60}$	72.57	62.29 $^{\uparrow 10.29}$	78.14	73.04$^{\uparrow 5.11}$	N/A	N/A
STGCN-Transformer	42.25	42.44 $^{\downarrow 0.19}$	50.67	51.87 $^{\downarrow 1.20}$	63.35	61.86 $^{\uparrow 1.49}$	71.89	73.00$^{\downarrow 1.11}$	11.32	9.39$^{\uparrow 1.93}$
SLGCN-Transformer	35.85	**34.69**$^{\uparrow 1.16}$	44.99	**42.74**$^{\uparrow 2.25}$	59.43	**55.06**$^{\uparrow 4.37}$	**66.68**	67.25$^{\downarrow 0.57}$	**8.11**	**8.11** $^{0.00}$

recognition [11]. WLASL includes 2,000 common ASL words, and it contains four sub-dataset depending on the number of words. We report the performance for each sub-dataset in this paper.

Another dataset is a JSL video dataset, including both isolated and continuous words [4]. The signers included 37 adults with experience in Japanese sign language. The dataset included 275 types of isolated word videos and 120 types of sentence videos. The vocabulary was conversational, reflecting topics that are spoken in a common city office. The videos were recorded at 30 frames per second with 640×360 pixels.

Fig. 3 shows examples of sign language videos. Fig. 3 (a) and (b) show examples of isolated word videos expressing "Nyuuseki" and "Kibou", respectively. "Nyuuseki" and "Kibou" mean "registration of marriage" and "hope", respectively. Fig. 3 (c) is an example of a sentence video. The sentence consists of "Nyuuseki" and "Kibou", which means "I'd like to register our marriage." "Arm up" and "Arm down" are transition motions between "Short pause" and word motions. "Transition" indicates a transitional motion between word motions. These marginal motions do not convey lexical meaning. Therefore, they were not included in the target words $\boldsymbol{L}$.

Fig. 3 shows that the signers posed in a static posture at the beginning and the end of a sign. The frames between these static postures are defined as a single action instance.

Table I shows a summary of the JSL dataset. We note that horizontally flipped tracking sequences were added to avoid effects related to signers' dominant hands. The number of action types is shown in parentheses. The sentences consisted of 200 types of words, and 42 types of words were only included in the sentences. Therefore, the total vocabulary of the dataset comprised 320 words, including $<$start $>$, $<$end $>$, and $<$pad $>$. We selected two signers for the test because the dataset included a bias in the number of videos for each signer. The largest number of videos were recorded for these two test signers.

Owing to the limitation of batch training, the input feature sequences are required to have a fixed length both during training and testing. Similarly, the input word sequences are also required to have a fixed length during training. Let T_{max} and S_{max} be the maximum lengths of the input feature and word sequences in the dataset, respectively. If the length of an input feature sequence is $T < T_{max}$, $\mathbf{0} \in \mathcal{R}^{54}$ is inserted after $\boldsymbol{x}_T$. Similarly, if the length of a sequence is $S < S_{max}$, $< pad >$ is inserted after y_S. The maximum numbers of frames were 247 and 578 for WLASL and JSL datasets, respectively. Therefore, we applied padding with $T_{max} = 247$ and $T_{max} = 578$ to the input feature sequences of WLASL and JSL datasets, respectively. Similarly, the output word sequences of the JSL dataset were padded with $S_{max} = 13$, including $<$start $>$ and $<$end $>$, because the maximum number of words was 11 for the dataset.

B. Recognition performance

The training settings are described as follows. The batch size was 32 in all training procedures performed. The learning rate is set as 0.0003, and the adaptive moment estimation [23] was applied to update the parameters. A categorical cross-entropy was used for loss function. 150 training epochs were used for all training procedures.

The best word error rates achieved during the training processes are shown in Table II. The superscripts indicate the performance improvement and depreciation. As shown in Table II, the model with MBN improve the performance in many cases. SLGCN with Transformer achieves the best performance in the isolated and continuous word recognition tasks. These results show the effectiveness of the proposed method and model design.

VI. Conclusions and Future Work

In this study, we have proposed masked batch normalization, which extends batch normalization to include masking of dummy signals. We applied masked batch normalization to GCN-based sign language recognition models. The experimental results using the two types of sign language video datasets designed for the isolated and continuous words recognition tasks described above show that the proposed method improves recognition models in many cases.

Although the combination with Transformer improves the performance, it tends to weaken the effectiveness of MBN. The current simple training pipeline might destabilize the complex model. We expect that generalization techniques may improve the model stability.

Batch normalization is among the standard techniques used in DNNs and has been adopted in many models. Although we focus on the padding of sequences in the present work, dummy signals are also generated by image data augmentation. We believe that masked batch normalization can improve many types of vision models incorporating the standard batch normalization.

We expect that the effect of dummy signals on normalization is also common in other derivative methods [21], [24], [25], [26], [27]. We plan to apply masked normalization to such derivative methods in the future.

REFERENCES

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

[2] O. Koller, N. C. Camgoz, H. Ney, and R. Bowden, "Weakly supervised learning with multi-stream cnn-lstm-hmms to discover sequential parallelism in sign language videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 9, pp. 2306–2320, Apr. 2020.

[3] N. C. Camgoz, O. Koller, S. Hadfield, and R. Bowden, "Sign language transformers: Joint end-to-end sign language recognition and translation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Jun. 2020, pp. 10 023–10 033.

[4] N. Takayama, G. Benitez-Garcia, and H. Takahashi, "Sign language recognition based on spatial-temporal graph convolution-transformer (in publishing)," vol. 87, no. 12, Dec. 2021.

[5] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. of the 32nd Int. Conf. on Machine Learning*, vol. 37, 2015, pp. 448–456.

[6] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. 32nd AAAI Conf. on Artificial Intelligence*, Feb. 2018, pp. 7444–7452.

[7] S. Jiang, L. Wang, Y. Bai, K. Li, and Y. Fu, "Skeleton aware multi-modal sign language recognition," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops*, Jun. 2021.

[8] K. Cheng, Y. Zhang, C. Cao, L. Shi, J. Cheng, and H. Lu, "Decoupling gcn with dropgraph module for skeleton-based action recognition," in *Proc. of the European Conference on Computer Vision, LNCS 12369*, 2020, pp. 536–553.

[9] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Skeleton-based action recognition with multi-stream adaptive graph convolutional networks," *IEEE Trans. Image Process.*, vol. 29, pp. 9532–9545, Oct. 2020.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, Dec. 2017, pp. 5998–6008.

[11] D. Li, C. Rodriguez, X. Yu, and H. Li, "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison," in *Proc. IEEE/CVF Winter Conf. on Applications of Computer Vision*, Mar. 2020, pp. 1459–1469.

[12] H. Cooper, N. Pugeault, and R. Bowden, "Reading the signs: A video based sign dictionary," in *IEEE Int. Conf. on Computer Vision Workshops*, Nov. 2011, pp. 914–919.

[22] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Jul. 2017, pp. 7291–7299.

[13] P. Tomas, C. James, and Z. Andrew, "Large-scale learning of sign language by watching tv (using co-occurrences)," in *Proc. British Machine Vision Conf.*, Sep. 2013, pp. 1–11.

[14] J. Forster, O. Koller, C. Oberdörfer, Y. Gweth, and H. Ney, "Improving continuous sign language recognition: Speech recognition techniques and system design," in *Proc. Fourth Workshop on Speech and Language Processing for Assistive Technologies*, Aug. 2013, pp. 41–46.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of the Int. Conf. on Learning Representations*, May 2015.

[16] C. Szegedy, W. Liu, Y. Jia, S. Pierre, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Jun. 2015, pp. 1–9.

[17] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. of the 36th Int. Conf. on Machine Learning*, ser. Proc. of Machine Learning Research, vol. 97, Jun. 2019, pp. 6105–6114.

[18] M. De Coster, M. Van Herreweghe, and J. Dambre, "Sign language recognition with transformer networks," in *Proc. 12th Language Resources and Evaluation Conf.*, May 2020, pp. 6018–6024.

[19] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Third Int. Conf. on Learning Representations*, Jul. 2015, pp. 1–15.

[20] N. Takayama and T. Hiroki, "Data augmentation using feature interpolation of individual words for compound word recognition of sign language," in *Proc. Int. Conf. on Cyberworlds*, Sep. 2020, pp. 137–140.

[21] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," arXiv preprint arXiv:1607.06450, 2016.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of the Third Int. Conf. on Learning Representations*, May 2015.

[24] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," arXiv preprint arXiv:1607.08022, 2017.

[25] B. Li, F. Wu, K. Q. Weinberger, and S. Belongie, "Positional normalization," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 1–13.

[26] S. Shen, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, "Powernorm: Rethinking batch normalization in transformers," in *Proc. of the 37th Int. Conf. on Machine Learning*, Jul. 2020, pp. 119:8741–8751.

[27] P. Luo, R. Zhang, J. Ren, Z. Peng, and J. Li, "Switchable normalization for learning-to-normalize deep representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 712–728, Feb 2021.