APSIPA Transactions on Signal and Information Processing, 2022, 11, e22

Original Paper

American Sign Language Fingerspelling Recognition in the Wild with Iterative Language Model Construction

Wuttipong Kumwilaisak[1], Peerawat Pannattee[1], Chatchawarn Hansakunbuntheung[2] and Nattanun Thatphithakkul[2*]

[1] *Department of Electronics and Telecommunication Engineering, King Mongkut's University of Technology Thonburi, Bangkok 10140, Thailand*
[2] *Assistive Technology and Medical Devices Research Center, National Science and Technology Development Agency, Pathum-Thani 12120, Thailand*

ABSTRACT

This paper proposes a novel method to improve the accuracy of the American Sign Language fingerspelling recognition. Video sequences from the training set of the "ChicagoFSWild" dataset are first utilized for training a deep neural network of weakly supervised learning to generate frame labels from a sequence label automatically. The network of weakly supervised learning contains the AlexNet and the LSTM. This trained network generates a collection of frame-labeled images from the training video sequences that have Levenshtein distance between the predicted sequence and the sequence label equal to zero. The negative and positive pairs of all fingerspelling gestures are randomly formed from the collected image set. These pairs are adopted to train the Siamese network of the ResNet-50 and the projection function to produce efficient feature representations. The trained Resnet-50 and the projection function are concatenated with the bidirectional LSTM,

*Corresponding author: Wuttipong Kumwilaisak, wuttipong.kum@kmutt.ac.th. This work was supported in part by King Mongkut's University of Technology Thonburi, and National Science and Technology Development Agency, Thailand.

Received 16 January 2022; Revised 13 May 2022
ISSN 2048-7703; DOI 10.1561/116.00000003

a fully connected layer, and a softmax layer to form a deep neural network for the American Sign Language fingerspelling recognition. With the training video sequences, video frames corresponding to the video sequences that have Levenshtein distance between the predicted sequence and the sequence label equal to zero are added to the collected image set. The updated collected image set is used to train the Siamese network. The training process, from training the Siamese network to the update of the collected image set, is iterated until the image recognition performance is not further enhanced. The experimental results from the "ChicagoFSWild" dataset show that the proposed method surpasses the existing works in terms of the character error rate.

Keywords: Fingerspelling recognition, weakly supervised learning, iterative training, deep learning, Siamese network.

1 Introduction

Sign language recognition is a challenging problem in video processing because sign language video content consists of fine details of complex hand postures and movements. Fingerspelling is a subtype of sign language that uses letters from the writing system to spell proper nouns and new emergent vocabularies that do not yet have corresponding signs. Although fingerspelling recognition is a subset of sign language recognition, it involves finer details of finger postures with fast movements and a high degree of similarity among fingerspelling motions.

Due to the difficulties of precise fingerspelling recognition, several automatic fingerspelling recognition methods [2, 5, 6, 21–24, 31, 32] have been proposed on environment-controlled datasets. An early, large, and most-common fingerspelling dataset for fingerspelling recognition is the American Sign Language (ASL) Fingerspelling dataset, with over 64,000 hand-area fingerspelling images and their depth images collected by Microsoft Kinect [21]. The dataset was utilized together with a multi-class random forest in [21] to classify fingerspelling gestures. Since then, the ASL Fingerspelling dataset has become a widely used dataset to evaluate proposed fingerspelling recognition systems in [5, 6, 23, 24, 31, 32]. After the emergence of deep learning applications in image and video recognition, a convolutional neural network (CNN) [7, 14, 17, 20, 29] has been widely adopted to solve the fingerspelling recognition problem [2, 22, 28]. In [2], a shallow CNN was utilized for ASL fingerspelling recognition. Inputs of this CNN are YUV color components and depth information of an image. However, the recognition performance was not significantly improved from the traditional methods. A deeper CNN based on the Inception-ResNet [3, 4, 13]

was adopted to recognize hand gestures in [22]. With a more powerful neural network, the recognition performance from [22] surpasses previous techniques. However, the algorithm was limited to only hand-cropped fingerspelling images. The research work in [28] proposed the fingerspelling recognition architecture, which deployed the depth-aware attention between color components and depth to automatically focus on the key fingers and hand regions. Its recognition results outperformed previous works. However, the proposed algorithm was limited to still images and might not be applicable to real-time applications.

Fingerspelling recognition in video sequences has received attention recently. The research work in [15] utilized the Histograms of Oriented Gradients (HOG) to derive handcraft features of each video frame. However, it needs training video sequences with frame-level labels, which may not be available. In [18, 25], the autoencoder and the attention module were deployed to automatically generate frame labeling from sequence labeling in order to achieve better fingerspelling recognition accuracy. However, these methods were restricted to controlled and set up environments. They might not perform well under naturally taken video sequences. Recently, the video data set called "ChicagoFSWild" dataset [27] has collected naturally taken ASL fingerspelling video sequences from the Internet. The environments of video sequences in this dataset are dynamic and reflect actual use cases. Figure 1 shows video frame examples from this dataset. Based on the "ChicagoFSWild" dataset, the work in [27] deployed the Faster R-CNN object detector [9] to detect hand areas. Then, it utilized the AlexNet [1] together with the LSTM and Connectionist Temporal Classification (CTC) [10] to obtain a recognized letter sequence. The improved version of [27] was proposed in [26] by a new hand cropping algorithm and a spatial attention module to better crop hand areas in a video frame. This new hand cropping algorithm was deployed to improve fingerspelling recognition performance by incorporating hand skeletal information and hand cropped frames in [19]. Furthermore, The transformer encoder replaced the LSTM in [8] to better extract temporal correlation between video frames. This modification enhanced the recognition performance from the previous methods [26, 27]. However, the recognition performance of these methods in terms of the Character Error Rate (CER) was still not satisfied. Moreover, the architectures of these previous works required the whole video sequence as an input to learn both spatial and temporal simultaneously. As a result, these architectures consumed significant computational resources during the training process.

This paper proposes a novel method to improve the accuracy of the ASL fingerspelling recognition. Video sequences from the training set of the "ChicagoFSWild" dataset are first utilized for training a deep neural network of weakly supervised learning to generate frame labels from a sequence label automatically. The network of weakly supervised learning contains the AlexNet and the LSTM. This trained network generates a collection of frame-labeled images

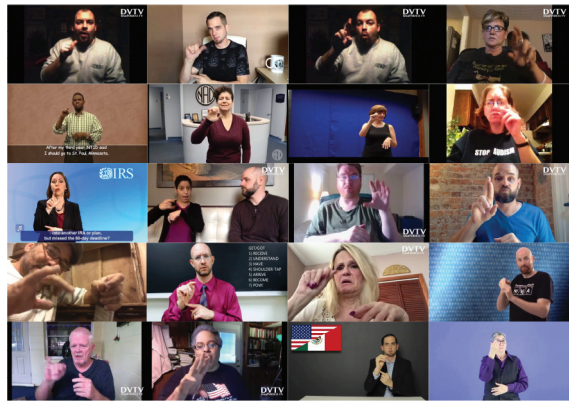

Figure 1: Video frame examples from the "ChicagoFSWild" dataset.

from the training video sequences that have Levenshtein distance between the predicted sequence and the sequence label equal to zero. The negative and positive pairs of all fingerspelling gestures are randomly formed from the collected image set. These pairs are adopted to train the Siamese network of the ResNet-50 and the projection function to produce efficient feature representations. The trained Resnet-50 and the projection function are concatenated with the bidirectional LSTM, a fully connected layer, and a softmax layer to form a deep neural network for the ASL fingerspelling recognition. With the training video sequences, video frames corresponding to the video sequences that have Levenshtein distance between the predicted sequence and the sequence label equal to zero are added to the collected image set. The updated collected image set is used to train the Siamese network. The training process, from training the Siamese network to the update of the collected image set, is iterated until the image recognition performance is not further enhanced. The experimental results from the "ChicagoFSWild" dataset show that the proposed method surpasses the existing works in terms of the character error rate. The contributions of this paper can be summarized as follows.

1. We propose using weakly supervised learning to automatically generate a frame-label image dataset for training a deep neural network of the ASL fingerspelling recognition.

2. We propose a training method of a very deep neural network based on the Siamese network-based feature embedding that can avoid using significant computational resources during the training process and provide efficient feature representations.

3. We introduce an iterative method in constructing a deep neural network

for the ASL fingerspelling recognition by updating an image set used to train the Siamese network.

This paper is organized as follows. Section 2 describes the architectural overview of our proposed ASL fingerspelling recognition system. Section 3 presents the method of automatic frame labeling based on weakly-supervised learning. Section 4 explains a new training method for a deep neural network that relied on Siamese network-based feature embedding. Section 5 describes the iterative construction of the ASL fingerspelling recognition model. Experimental results are presented in Section 6. The concluding remarks are in Section 7.

2 Architectural Description

This section provides the architectural description of the proposed ASL fingerspelling recognition method. The objective of our technique is to derive a character sequence $\mathbf{U}$ from an ASL fingerspelling video sequence $\mathbf{X}$. Figure 2 illustrates the deep learning architecture of the recognition system. To train our recognition system, a training video sequence $\mathbf{X}$ from a training set together with its sequence label $\mathbf{Q}$ is utilized. We should notice no specified boundaries among fingerspelling gestures in a training video sequence, which poses a significant challenge in recognizing multiple gestures. The "ChicagoFSWild" dataset is utilized in both the training and testing process since all video sequences in the dataset are naturally taken under dynamic environments and reflect real use cases.

Let us consider the top portion of Figure 2. Video sequences are preprocessed with the iterative attention module. The iterative attention module will automatically crop image areas with hand movement. Since training video sequences are labeled at a sequence level, we feed cropped video frames together with a sequence label to the trained weakly supervised learning network as shown in the middle portion of Figure 2. We should emphasize that the weakly supervised learning module is trained previously before deployment with the training set of the "ChicagoFSWild" dataset. The weakly supervised learning will output video frames with their labels. The weakly supervised learning employs the CTC [10] to align a sequence label to video frames. Video frames corresponding to video sequences with one-hundred percent recognition accuracy are included in a frame label dataset Θ. The recognition accuracy is measured using the Levenshtein distance [30] converted to the Character Error Rate (CER).

Our ASL fingerspelling recognition consists of the ResNet-50, the projection function, and bidirectional LSTM. Training such a deep neural network requires significant computational resources. As a result, we train a deep neural network

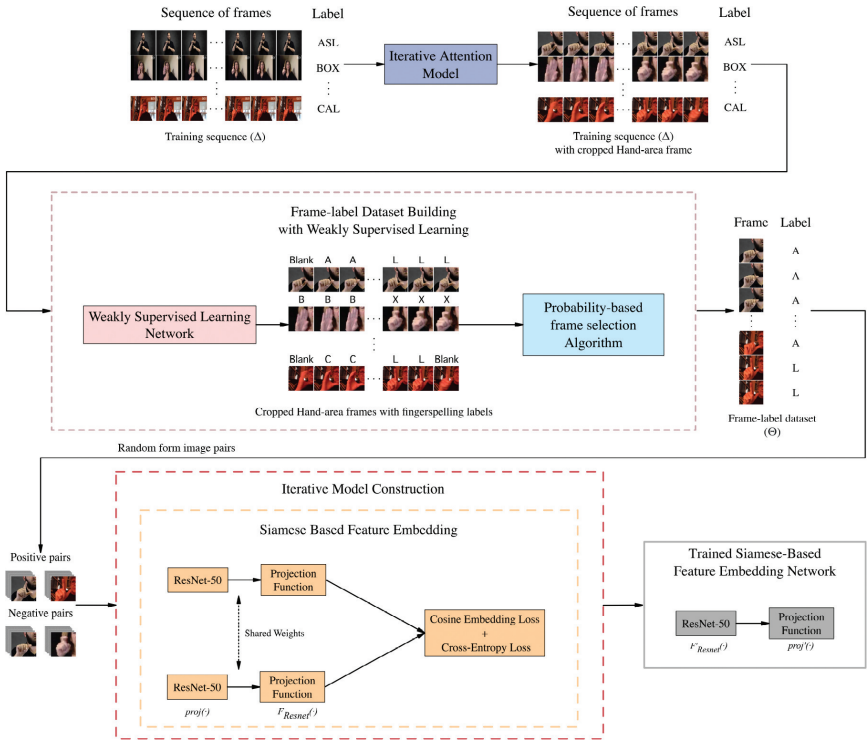

Figure 2: The overview of the proposed method's training process.

using an alternative method based on the Siamese network architecture. Let us consider the bottom portion of Figure 2. Based on Θ, we randomly form positive and negative pairs of all image gestures. These image pairs are fed to train the Siamese network. The Siamese network contains the ResNet-50 and the projection function. Both of them extract spatial features that existed in video frames. The loss function of the Siamese network is a combination of cosine embedding and cross-entropy loss functions. It is designed to give good feature vector representation resulting in better gesture classification. After training, we form a recognition network. It consists of the ResNet-50 and the projection function concatenated with a bidirectional LSTM, a fully connected layer, and a softmax layer. We need to train a bidirectional LSTM, a fully connected layer, and a softmax layer with video sequences in a training set. This can be done by fixing weights of the ResNet-50 and the projection function. Then, only weights of the bidirectional LSTM and a fully connected layer are updated. Finally, we have all trained components in a recognition network. Training video sequences from the training set are fed to our recognition network. Video frames corresponding to the perfect

Table 1: Summary of notations.

$\mathbf{X}$	:	the training video sequence
$\mathbf{Q}$	:	the ground-truth character sequence of the training video sequence $\mathbf{X}$
$\mathbf{U}$	:	the predicted character sequence of the training video sequence $\mathbf{X}$
Δ	:	the training set of the "ChicagoFSWild" dataset
Θ	:	the frame label dataset
$\boldsymbol{\pi}$	:	the predicted sequence of frame labels of the training video sequence $\mathbf{X}$
$\boldsymbol{\pi}^*$	:	the most probable predicted sequence of frame labels of the training video sequence $\mathbf{X}$
α_t	:	the most probable fingerspelling gesture corresponding to frame x_t
$\mathcal{B}$	:	the many-to-one mapping function that map $\boldsymbol{\pi}$ to $\mathbf{U}$
$\mathcal{V}$	:	the set of fingerspelling gestures
$\mathcal{V}'$	:	the set of fingerspelling gestures include "blank" label

recognition are included in Θ. The Siamese network is iteratively trained until the recognition performance is not further improved. Table 1 lists the notations that we will use in this paper.

3 Automatic Frame Labeling with Weakly Supervised Learning

In this section, we deploy a neural network architecture from Shi *et al.* [26], as illustrated in Figure 3, to automatically label video frames within the original sequence-labeled fingerspelling video data set Δ. The AlexNet with attention model [26] is used to extract spatial features of each video frame. Define $\mathbf{X} = \{x_t\}_{t=1}^{T}$ be an input sequence from frame one to frame T, where x_t is the t^{th} frame in the sequence. Let function $F_{\text{AlexNet}}(\cdot)$ be a function to extract spatial feature s_t of frame t with AlexNet, the spatial feature s_t can be expressed as

$$s_t = F_{\text{AlexNet}}(x_t). \tag{1}$$

s_t is further fed to an attention cell. An output of the attention cell is an attention map expressed as β_t. β_t provides essential location information in the spatial feature map employed for fingerspelling recognition. The computation of $\beta_t(i, j)$, which is the attention value at position (i, j), can be

expressed as

$$v_t = W \cdot \tanh(s_t), \tag{2}$$

$$\beta_t(i,j) = \frac{\exp(v_t(i,j))}{\sum_{\forall m,n} \exp(v_t(i,j))}, \tag{3}$$

where W is a weight matrix. Moreover, in high movement video data, the optical flow-based attention of frame t, M_t, can be included to improve the knowledge of essential regions. The spatial feature with attention at frame t can be computed as

$$a_t = \frac{s_t \otimes \beta_t \otimes M_t}{\sum_{\forall p,q} \beta_t(p,q) M_t(p,q)}, \tag{4}$$

where $\otimes$ is an element-wise multiplication, and $\beta_t(p,q)$ and $M_t(p,q)$ are the values of β_t and M_t at position (p,q).

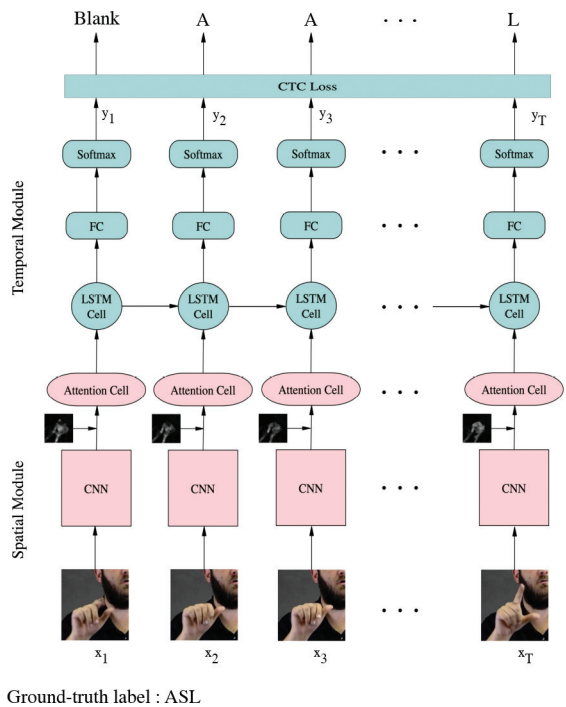

Figure 3: The architecture of the weakly supervised learning network for automatically frame labeling.

To exploit the temporal correlation between spatial features from video frames, a Long-Short Term Memory (LSTM) network [12] is used. Let y_t be a

probability vector of all possible fingerspelling gestures at frame t. It can be reckoned via

$$e_t = \sum_{i,j} a_t(i,j), \tag{5}$$

$$h_t, c_t = F_{\text{LSTM}}(h_{t-1}, c_{t-1}, e_t), \tag{6}$$

$$y_t = \text{SM}(W \cdot h_t + b), \tag{7}$$

where $a_t(i,j)$ is an attention map value of frame t at position (i,j). h_t and c_t are hidden and cell states of the LSTM network of frame t. $F_{\text{LSTM}}(\cdot)$ is a function derived from the LSTM network to extract temporal information. $\text{SM}(\cdot)$ is a softmax classifier. W and b are the weight and bias of the Softmax classifier. As we previously described, only sequence-label information is available from the video data set Δ. As a result, the CTC [10] is deployed to produce a frame label for each video frame. The CTC will label a video frame as a "blank" when it can not classify a gesture of the video frame. In general, the recognized gestures correspond to transition video frames between consecutive meaningful gestures.

Next, let us formulate a framework of the automatic frame labeling task. Let $P\{\mathbf{Q} \mid \mathbf{X}\}$ be the conditional probability of sequence label $\mathbf{Q} = \{q_1, q_2, \ldots, q_L\}$ given an input sequence $\mathbf{X}$ and L is the length of a sequence label. Then,

$$P\{\boldsymbol{\pi} \mid \boldsymbol{X}\} = \prod_{t=1}^{T} P\{\pi_t \mid \boldsymbol{X}\} = \prod_{t=1}^{T} y_{t,\pi_t}, \tag{8}$$

$$P\{\boldsymbol{Q} \mid \boldsymbol{X}\} = \sum_{\forall \pi \in \mathcal{B}^{-1}(\boldsymbol{q})} P\{\boldsymbol{\pi} \mid \boldsymbol{X}\}, \tag{9}$$

where $\boldsymbol{\pi}$ is a predicted frame-label sequence of a sequence $\boldsymbol{X}$, π_t is a frame label of the i^{th} frame in the sequence, which $\pi_t \in \mathcal{V} \cup$ "*blank*". $\mathcal{V}$ is a set of fingerspelling gestures, consists of characters "A"-"Z" and symbols "@", "&", "." as well as punctuate. $\mathcal{B}$ is a many-to-one mapping which simply removes all "blank" label and repeated labels from the frame-labeled sequence $\boldsymbol{\pi}$. For example, $\mathcal{B}(\_cc\_aat\_t\_) = \mathcal{B}(c\_aa\_t) = cat$, where "_" represents the "blank" label. $\mathcal{B}^{-1}(q)$ is the inverse mapping function of $\mathcal{B}$, where $\mathcal{B}^{-1}(q) = \{\boldsymbol{\pi} \mid \mathcal{B}(\boldsymbol{\pi}) = q\}$.

We adopt the CTC loss function to train the neural network as [10]

$$\mathcal{L}_{\text{CTC}} = -\log(P\{\boldsymbol{Q} \mid \boldsymbol{X}\}). \tag{10}$$

Given a sequence $\boldsymbol{X}$, the most probable predicted sequence of frame labels can be determined from

$$\boldsymbol{\pi}^* = \arg\max_{\forall \pi} P\{\boldsymbol{\pi} \mid \boldsymbol{X}\}, \tag{11}$$

from $\boldsymbol{\pi}^*$, the most probable fingerspelling gesture corresponding to frame x_t and the predicted character sequence can be computed via

$$\alpha_t = \arg \max_{\forall \pi^* \in \mathcal{V}'} P\{\pi_t^* \mid \boldsymbol{X}\}, \tag{12}$$

$$\boldsymbol{U} = \mathcal{B}(\boldsymbol{\pi}^*), \tag{13}$$

where $\mathcal{V}' = \mathcal{V} \cup$"*blank*." α_t and $\boldsymbol{U}$ are the most probable fingerspelling gesture corresponding to frame x_t and the predicted character sequence, respectively. The procedure of the automatic frame labeling with weakly supervised learning can be summarized in Algorithm 1.

Algorithm 1 The algorithm description of the automatic frame-labeled generation.

Train the weakly supervised learning network based on Equation 10
for $\mathbf{X}$, $\mathbf{Q} \in \Delta$ **do**
  Run inference on $\mathbf{X}$ with the trained weakly supervised learning network in Equation 13 to obtain the predicted character sequence $\boldsymbol{U}$
  Compute the Levenshtein distance between $\boldsymbol{U}$ and $\mathbf{Q}$
  if The Levenshtein distance is equal to zero **then**
    for $x_t \in \mathbf{X}$ **do**
      Solve Equation 12 to obtain α_t
      if α_t ! ="blank" **then**
        $\Theta \leftarrow (x_t, \alpha_t)$
      end if
    end for
  end if
end for
return Θ

4 Training Deep CNN with Siamese Network-Based Feature Embedding

Training a deep CNN to recognize fingerspelling gestures might consume gigantic computational resources. We need to incorporate many video frames as inputs of the CNN to obtain both spatial and temporal features. If computational resources are not available, utilizing a deep CNN to yield correct recognized gestures may be prohibited. As a result, we propose an alternative training technique that relies on Siamese-Network-based feature embedding requiring much less training resources. The ResNet-50 networks [11, 16] are used in a Siamese network. The Siamese network aims to construct feature

vectors representing different gestures. Ideally, if two images are labeled as the same fingerspelling gesture, these two images' feature vectors extracted from the Siamese network should be close. In contrast, feature vectors from different fingerspelling gestures should have significant distances. We randomly form positive and negative pairs of all gestures based on Θ constructed in the previous section. Images of each positive pair have the same frame label, whereas each negative pair has different frame labels. These pairs are used to train the Siamese network to improve the extracted spatial feature. The structure of the Siamese network is shown in Figure 4.

Let I_1 and I_2 be two input images feeding to the Siamese network. $F_{Resnet}(\cdot)$ is a function derived from the Siamese network, which extracts image features. Suppose that f_1 and f_2 are features obtained from the ResNet-50. In other words,

$$f_1 = flat(F_{\text{Resnet}}(I_1)), \tag{14}$$

$$f_2 = flat(F_{\text{Resnet}}(I_2)), \tag{15}$$

where $flat(\cdot)$ is a flatten function changing two-dimensional signals to one-dimensional signals. For example, in this paper, our image has dimensions 244×244. The output from the ResNet-50 and the flatten function will have dimensions 2048. We project this one-dimensional vector to a vector with less dimension via

$$v_1 = proj(f_1), \tag{16}$$

$$v_2 = proj(f_2), \tag{17}$$

where $proj : R^r \rightarrow R^d$ is a projection function from a vector with dimension r to a vector with dimensional d. In this paper, we set $r = 2048$ and $d = 1024$.

Next, we formulate the loss function used to train the Siamese network. First, we adopt two loss functions. The first loss function is the cosine embedding loss function [11]. The objective of this cost function is to force a pair of feature vectors obtained from the Siamese network to be close if this image pair comes from the same gesture. The cosine embedding loss function can be expressed as

$$\mathcal{L}_E(v_1) = \begin{cases} 1 - \frac{v_1 \cdot v_2}{||v_1|| ||v_2||}, & y = 1, \\ \max(0, \frac{v_1 \cdot v_2}{||v_1|| ||v_2||}), & y = -1, \end{cases} \tag{18}$$

where $y = 1$ and $y = -1$ represent the positive and negative pairs of fingerspelling gesture images, respectively. The second loss function is introduced to measure the recognition accuracy. We first classify a reference gesture image I_1 with a softmax classifier, which yeilds

$$\gamma_1'(I_1) = \text{SM}(W_r \cdot v_1 + b_r), \tag{19}$$

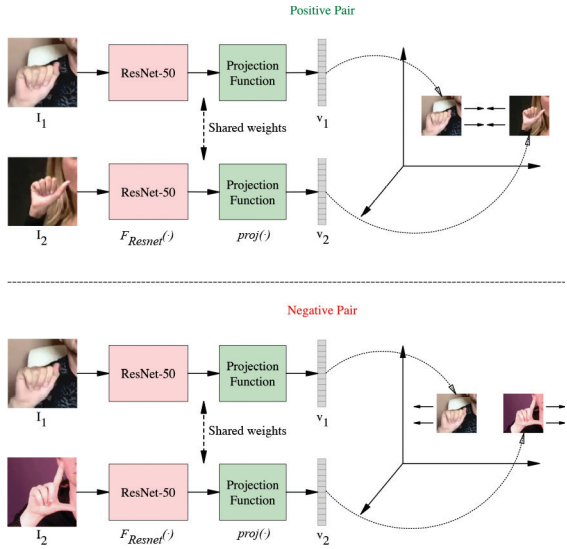

Figure 4: The architecture of the Siamese network for feature embedding.

where W_r and b_r are a weight matrix and a bias vector of the softmax classifier, respectively. Suppose that $\gamma(I_1)$ is a one-hot vector representing gesture class image I_1 belongs. We have 26 classes for fingerspelling gestures representing the letter "A" to the letter "Z." Hence, $\gamma_1'(I_1)$ and $\gamma_1(I_1)$ will have dimensions of 26×1. The cross-entropy loss function is deployed as the second loss function, which is

$$\mathcal{L}_{\mathrm{CE}}(v_1) = -\sum_{i=1}^{26} \gamma(i) \log(\gamma'(i)), \tag{20}$$

where $\gamma(i)$ and $\gamma'(i)$ are the i^{th} components of γ and γ', respectively. We combine two loss functions to obtain the final loss function as

$$\mathcal{L}_V(v_1) = \mathcal{L}_E(v_1) + \mathcal{L}_{\mathrm{CE}}(v_1). \tag{21}$$

The Siamese network will be trained with the combined loss function to obtain improved image features and enhance recognition accuracy.

5 Fingerspelling Recognition with Iterative Model Construction

To enhance the performance of our proposed fingerspelling recognition method, for each training session of the Siamese network, the iteratively trained ResNet-50 and the projection function are deployed to recognize fingerspelling gestures.

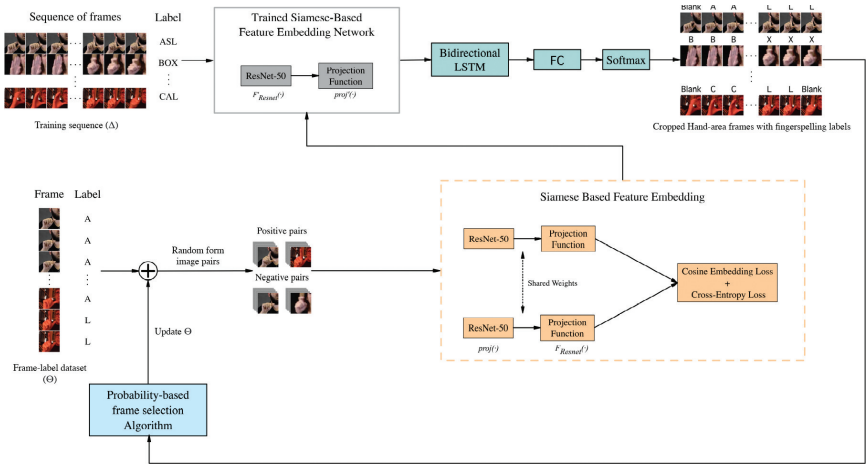

Figure 5: Iterative model construction.

Let us consider the process of the iterative training in Figure 5. From training video sequences in Δ, we apply the ResNet-50 and the projection function to extract the spatial features of video frames. Suppose that video sequence $\mathbf{X}$ from a training dataset is fed to train our recognition model. We can obtain feature vector v'_t of frame x_t within $\mathbf{X}$ via

$$f'_t = flat(F'_{\text{Resnet}}(x_t)), \tag{22}$$

$$v'_t = proj'(f'_t), \tag{23}$$

where $F'_{\text{Resnet}}(\cdot)$ and $proj'(\cdot)$ are the trained Resnet-50 network and the trained projection function from the Siamese network, and f'_t is a spatial feature of frame t. Then, temporal features are extracted from these spatial features among video frames using the bidirectional LSTM and are fed to a fully connected layer and a softmax layer, which can be expressed as

$$\overrightarrow{h_t}, \overrightarrow{c_t} = F_{\text{LSTM}}(\overrightarrow{h_{t-1}}, \overrightarrow{c_{t-1}}, v'_t), \tag{24}$$

$$\overleftarrow{h_t}, \overleftarrow{c_t} = F_{\text{LSTM}}(\overleftarrow{h_{t+1}}, \overleftarrow{c_{t+1}}, v'_t), \tag{25}$$

$$y_t = \text{SM}(W \times CONCAT(\overrightarrow{h_t}, \overleftarrow{h_t}) + b), \tag{26}$$

where $\overrightarrow{h_t}$ and $\overrightarrow{c_t}$ as well as $\overleftarrow{h_t}$ and $\overleftarrow{c_t}$ are the hidden and cell states of the LSTM corresponding to frame t in the forward and backward directions, respectively. W and b are the weight and a bias of the Softmax classifier. We train the bidirectional LSTM with CTC loss function as shown in Equation 10. Since we fully know sequence labels of video sequences from Δ, the frame labels obtained from a softmax layer can be verified. For example, suppose a sequence label of

a video sequence is "BOX" and the frame labels obtained from our recognition model are "B,B,O,X,X,X." In that case, we can declare that the recognition rate is 100 percent accurate. In this work, we adopt the Levenshtein distance [30] to measure the difference between the sequence label and the recognition results. If the Levenshtien distance between these two strings are equal to zero, we include all video frames within a video sequence together with their frame labels to Θ except video frames corresponding to "blank." Then, the Siamese network is trained with a new set of Θ. The model construction will be iterated until the training recognition results are not improved any more. The algorithm description of the iterative model construction can be expressed in Algorithm 2.

Algorithm 2 The algorithm description of the iterative model construction.

for i $\leftarrow$ 1 to Iteration **do**
  Train the Siamese network with Θ based on Equation 21 to obtain $F'_{Resnet}(\cdot)$ and $proj'(\cdot)$
  Extract spatial feature of each frame in Δ via Equation 23
  Train the spatial features with bidirectional LSTM via Equation 10
  for X, **Q** $\in \Delta$ **do**
    Run inference on **X** with the trained network in Equation 13 to obtain the predicted character sequence $\boldsymbol{U}$
    Compute the Levenshtein distance between $\boldsymbol{U}$ and **Q**
    if the Levenshtein distance is equal to zero **then**
      for $x_t \in$ **X do**
        Solve Equation 12 to obtain α_t
        if α_t != "blank" and $x_t \notin \Theta$ **then**
          $\Theta \leftarrow (x_t, \alpha_t)$
        end if
      end for
    end if
  end for
end for

Finally, the iteratively trained ResNet-50 and the projection function are deployed in our fingerspelling recognition system, as shown in Figure 6. As illustrated in Figure 6, the test video sequence is fed to the iterative attention model. The iterative attention model will automatically zoom in and focus on the image areas corresponding to hand gestures. The Resnet-50, the projection function, and the bidirectional LSTM are used to extract spatial and temporal features as described above. The initial output obtained from the softmax classification is a frame-level recognition result of a video sequence, which can

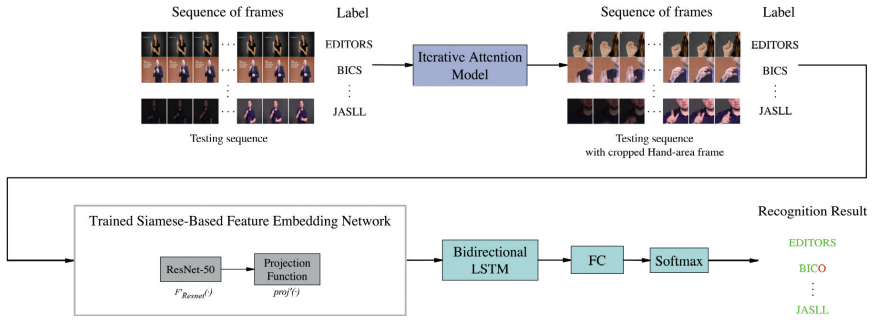

Figure 6: The deployment of the proposed ASL fingerspelling recognition system.

be expressed as

$$\boldsymbol{\pi}^* = \arg\max_{\forall \pi} P\{\boldsymbol{\pi} \mid \boldsymbol{X}\}, \tag{27}$$

where $\boldsymbol{\pi}^*$ is the most probable frame-level recognition result. The sequence of frame-level recognition is then mapped to the sequence-level recognition using the many-to-one mapping $\mathcal{B}$, which can be written as

$$\boldsymbol{U} = \mathcal{B}(\boldsymbol{\pi}^*). \tag{28}$$

We should notice that in practice the mapping $\mathcal{B}$ actually deletes the recognized letter corresponding to "blank" and the repetitions of recognized letters within $\boldsymbol{\pi}^*$.

6 Experimental Results

6.1 Dataset and Data Preprocessing

To assess the proposed ASL fingerspelling recognition system in challenging environments, we adopt a video dataset called "ChicagoFSWild" [27]. Video sequences in this dataset are not in controlled environments. They are most naturally taken from either good set-up cameras or mobile devices. Video sequences in the ChicagoFSWild are divided to be three sets: a training set (5455 sequences from 87 signers); a development set (868 sequences from 37 signers); and a test set (981 sequences from 36 signers). Each video set has no overlapping signers. Each video sequence is labeled at a sequence level. In our settings, Δ is the same as the training set of ChicagoFSWild. Θ is an image dataset firstly constructed from weakly supervised learning. Then, it is used for training the Siamese network and is updated during the iterative model construction. Our paper's development and test sets are the same as those of the ChicagoFSWild.

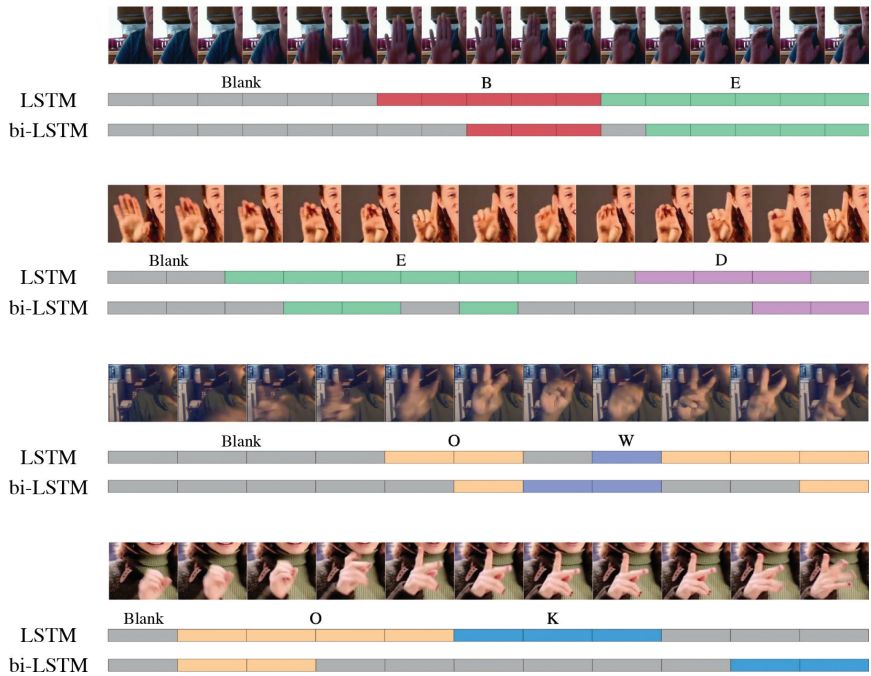

Figure 7: The comparison of the alignment result between Bidirectional LSTM and LSTM learned by CTC.

Each image is fed to the iterative attention module [26] to crop hand regions automatically. The attention module is trained with 30 epochs before deployment for three iterations. The zoom factors of the first, the second, and the third iterations are 0.72, 0.72, and 0.81, respectively. Figure 8 shows the hand cropping results obtained from the iterative attention module.

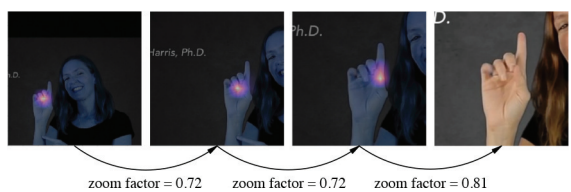

Figure 8: Hand cropping using iterative attention method.

We utilize the Levenshtien distance [30] to measure the difference between the actual sequence label and the recognition result. The Levensthtien distance can be converted to the character error rate (CER) via

$$\mathrm{CER} = \frac{i+s+d}{n}, \tag{29}$$

where i, s, and d are the minimal number of character insertions, substitutions, and deletions that transform the actual sequence label to the recognition result. n is the total number of characters in the sequence label.

6.2 Automatic Frame Labeling with Weakly Supervised Learning

This section assesses the performance of weakly supervised learning in automatic frame labeling. We investigate the uses of different deep neural network architectures in the weakly supervised learning architecture. We train the weakly supervised learning network for 40 epochs with the training set (Δ) and batch size one. The learning rate for the first 30 epochs is equal to 0.01, whereas the learning rate for the last ten epochs is equal to 0.001. The stochastic gradient descent is employed to be the optimizer. The AlexNet with attention module and the ResNet-18 are investigated for spatial feature extraction. In this study, both CNN architectures are used together with LSTM for temporal feature extraction. Table 2 shows a number of video frames in Θ obtained from the weakly supervised learning. We found that the AlexNet with attention module can provide more images in Θ and a better recognition (less CER) than the ResNet-18. The attention module plays a crucial role in improving the recognition result of the AlexNet even though the AlexNet's architecture is less complex than the ResNet-18.

Table 2: The recognition performance when different CNN architectures are employed.

CNN architecture	Amount of data in Θ	$1 - CER$ (Percent)
AlexNet with the attention module	31,228	45.4
ResNet-18	17,123	38.2

Table 3: The recognition performance when different LSTM architectures are employed.

LSTM Type	Amount of data in Θ	$1 - CER$ (Percent)
Standard LSTM	31,228	45.4
Bidirectional LSTM	28,739	46.9

Next, we study the effects of different Recurrent Neural Networks in the weakly supervised learning architecture. The one-directional and two-directional LSTMs are examined. The AlextNet with the attention module is used in this study. Table 3 concludes the experimental results. We found that the bidirectional LSTM provides better final fingerspelling recognition than the standard LSTM, giving fewer images in Θ. The reason behind

Figure 9: The recognition results in the frame level produced by the weakly supervised learning network compared with the results from a non-native sign language user.

this phenomenon may be based on the fact that the one-directional LSTM provides longer alignment between a sequence label and video frames than the bidirectional LSTM, as illustrated in Figure 7.

Figures 9 and 10 show the recognition results in a frame-level produced by the weakly supervised learning network compared with the results from humans who are not native sign language users, and the examples of images in Θ obtained from the weakly supervised learning network, respectively. The frame-level recognition results produced from the weakly supervised learning network are reasonably accurate and low noise, indicating that they can be utilized to represent frame-level labeling in the frame-label dataset Θ.

6.3 Siamese-based Feature Embedding

In this section, we investigate the performance of the proposed training method with Siamese-based feature embedding for the ASL fingerspelling recognition system. The Siamese network is trained for 15 epochs with the gradient descent optimizer. The batch size is 16. The learning rate for the first ten epochs is set to 0.01, whereas the learning rate of the last five epochs has a value of 0.001.

Figure 10: Examples of the selected frames in a frame-label dataset (Θ).

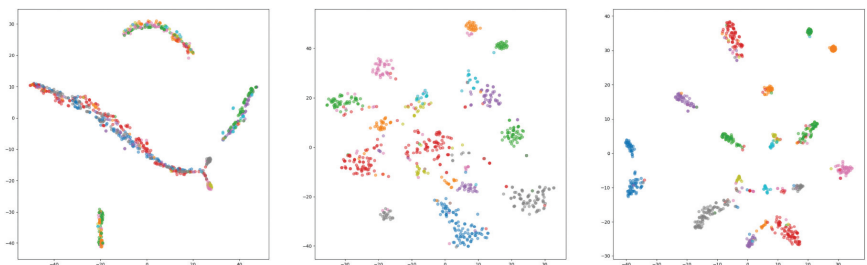

Figure 11: The feature vector projection in two dimensions with different loss functions: (a) cosine embedding; (b) cross-entropy; and (c) combined loss function.

Figure 11 shows the t-SNE plot of features obtained from the Siamese network with different loss functions. We can observe that the proposed combined loss function (cosine embedding and cross-entropy) can provide superior distinct features among different gestures to only cosine embedding or cross-entropy loss functions.

We examine the training performance of derived features when deploying them with different LSTM architectures. The LSTM architectures contain 512 nodes. We train the LSTM for ten epochs. The learning rate of the first seven epochs is set to 0.01, whereas the last three epochs possess a learning rate with a value of 0.001. Table 4 shows the recognition accuracy in terms of

Table 4: Training ASL fingerspelling recognition performance (1-CER) in percent when deploying the feature representation with different LSTM architectures.

	A number of layers		
LSTM architecture	1	2	3
LSTM	50.3	51.2	43.0
Bidirectional LSTM	51.6	50.2	50.7

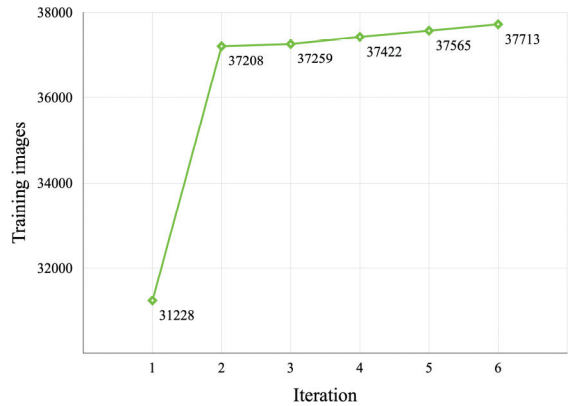

Figure 12: The increasing number of images in the training set of the Siamese network with iterations.

$1 - CER$. The bidirectional LSTM with one layer gives the best recognition result. Based on this verification, we adopt the one-layer bidirectional LSTM to our ASL fingerspelling recognition architecture.

6.4 Iterative Model Construction and Recognition

This section investigates the improved ASL fingerspelling recognition performance when employing iterative model construction. Figure 12 shows an increasing number of images in Θ in each iteration. The training of the Siamese network benefits from the increasing number of images. The development and test sets are used to evaluate our proposed ASL fingerspelling recognition method. Figure 13 illustrates the recognition accuracy in terms of $1 - \text{CER}$ for each iteration of iterative model construction when applying the recognition model to the development set. From iterations one to three, recognition performance is improved. However, after the third iteration, the recognition model becomes overfitting. As a result, the recognition accuracy worsens. Therefore, we early stop training our model with the iterative model construction at the third iteration.

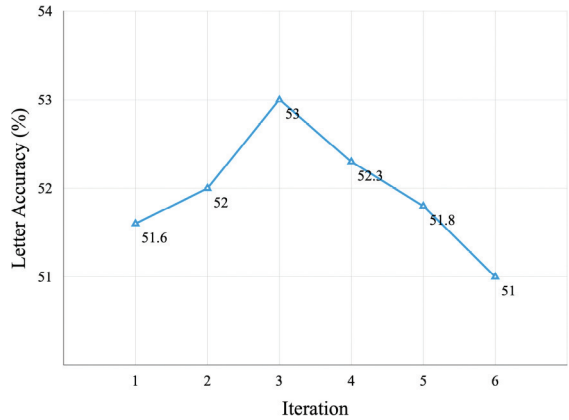

Figure 13: The ASL fingerspelling recognition accuracy with iterations.

Table 5: The ablation test results.

Method	$1 - CER$ (Percent)
Weakly supervised learning	45.4
Siamese-based feature embedding	51.6
Iterative model construction	53.0

Table 6: Fingerspelling recognition performance of the development set.

Method	$1 - CER$ (Percent)
Shi *et al.* [27]	42.8
Shi *et al.* [26]	46.8
Gajurel *et al.* [8]	46.9
The proposed method	53.0

Table 5 shows the ablation study on different components of our recognition architecture over the development set. The Siamese-based feature embedding can improve the recognition accuracy by around four percent from the weakly supervised learning. In addition, the iterative model construction can provide a two percent improvement of the recognition accuracy over that of the Siamese-based feature embedding. Tables 6 and 7 compare the recognition accuracy between the proposed method and the existing recognition techniques. All results are based on the ChicagoFSWild data set. It is apparent to see that our method can consistently surpass the existing techniques in both the development and the test sets.

Table 7: Fingerspelling recognition performance of the test set.

Method	$1 - CER$ (Percent)
Shi *et al.* [27]	41.9
Shi *et al.* [26]	45.1
Parelli *et al.* [19]	47.9
Gajurel *et al.* [8]	48.3
The proposed method	49.6

7 Conclusion

This paper presented a new training strategy for ASL fingerspelling recognition. The weakly supervised learning was first used to automatically generate an image dataset from the training video sequences. The images within the image dataset were utilized for training the Siamese network-based feature embedding to produce efficient feature representations. The trained ResNet-50 and the projection function obtained from the Siamese network are retrained with video sequences from the training set. However, only weights of the bidirectional LSTM and a fully connected layer are updated. Training video sequences are fed to a fully trained recognition network. Video frames corresponding to the video sequences that have Levenshtein distance between the predicted sequence and the sequence label equal to zero are added to the image dataset that will be deployed to train the Siamese network. The training of the Siamese network will be performed iteratively until the recognition performance is not further improved. The performance of the proposed deep neural network can achieve recognition accuracy of around 50 percent under naturally taken fingerspelling video sequences.

Biographies

Wuttipong Kumwilaisak received the B.E. degree from Chulalongkorn University, in 1995, the M.S. and Ph.D. degrees from the University of Southern California, in 2003, all in electrical engineering. From April 2003 to August 2004, he was a Senior Engineer and a Project Leader of the mobile platform solution team and multimedia laboratory at Samsung Electronics, Suwon, South Korea. He was a Postdoctoral Fellow with the Thomson Research Laboratory, Princeton, USA, from March 2006 to November 2006. He has been an Associate Professor with the Electronics and Telecommunication Department, King Mongkut's University of Technology Thonburi, Bangkok, Thailand. His current focused researches include multimedia communication, multimedia compression and processing, and deep learning. He is a member of IEEE.

Peerawat Pannattee received the B.E. degree in electronics and telecommunication engineering from King Mongkut's University of Technology Thonburi (KMUTT), Bangkok, Thailand, in 2019, where he is currently pursuing the M.E. degree in electrical engineering. His research interests include image processing, computer vision, deep learning.

Chatchawarn Hansakunbuntheung received the B.E. and M.E. degrees in electrical engineering from Chulalongkorn University, Thailand, in 1998 and 2000, respectively, and the Ph.D. degree in global information and telecommunication studies from the Waseda University, Tokyo, Japan, in 2010. He is currently a Researcher of the Accessibility and Assistive Technology Research Team, National Science and Technology Development Agency, Thailand. His research interests include speech and speaker recognition, speech synthesis, natural language processing, human machine interaction, and assistive technology.

Nattanun Thatphithakkul received his B.E and M.E degrees from Suranaree University, Thailand, in 2000 and 2002, respectively. He received his Ph.D. in computer engineering from King Mongkut's Institute of Technology Ladkrabang in 2008. He is now Chief of the Accessibility and Assistive Technology Research Team of the National Science and Technology Development Agency, Thailand. His research interests include speech and speaker recognition, speech synthesis, natural language processing, human-machine interaction and assistive technology.

References

[1] K. Alex, S. Ilya, and E. H. Geoffrey, "ImageNet Classification with Deep Convolutional Neural Networks", in *Advances in Neural Information Processing Systems (NIPS 2012)*, 25, 2012.

[2] S. Ameen and S. Vadera, "A Convolutional Neural Network to Classify American Sign Language Fingerspelling from Depth and Colour Images", *Expert System*, 34(3), 2017.

[3] S. Christian, I. Sergey, V. Vincent, and A. Alexander, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning", *AAAI Conference on Artificial Intelligence*, 31(1), 2017.

[4] S. Christian, V. Vincent, I. Sergey, S. Jon, and W. Zbigniew, "Rethinking the Inception Architecture for Computer Vision", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 2818–26.

[5] A. Dewinta and Y. Heryadi, "American Sign Language-based Finger-Spelling Recognition Using k-nearest Neighbors Classifier", in *2015 3rd International Conference on Information and Communication Technology (ICoICT)*, 2015, 533–6.

[6] B. Estrela, G. Camála-Chávez, M. Campos, W. Schwartz, and E. Nascimento, "Sign Language Recognition Using Partial Least Squares and RGB-D Information", in *IX Workshop de Visao Computacional (WVC).* 2013.

[7] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning Hierarchical Features for Scene Labeling", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 2013, 1915–29.

[8] K. Gajurel, C. Zhong, and G. Wang, "A Fine-Grained Visual Attention Approach for Fingerspelling Recognition in the Wild", arXiv:2105.07625, Aug. 2021.

[9] R. Girshick, "Fast R-CNN", in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, 1440–8.

[10] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data With Recurrent Neural Networks", in *23rd International Conference on Machine Learning (ICML)*, 2016, 369–76.

[11] K. He, X. Zhang, S. Ren, and S. J., "Deep Residual Learning for Image Recognition", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 770–8.

[12] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory", *Neural Computation*, 9, 1997.

[13] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian, "Deep Residual Learning for Image Recognition", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 770–8.

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale Video Classification with Convolutional Neural Networks", in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, 1725–32.

[15] T. Kim, J. Keane, W. Wang, H. Tang, J. Riggle, G. Shakhnarovich, D. Brentari, and K. Livescu, "Lexicon-free Fingerspelling Recognition from Video: Data, Models, and Signer Adaptation", *Computer Speech and Language*, 46(2), 2016, 209–32.

[16] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese Neural Networks for One-shot Image Recognition", in *32nd International Conference on Machine Learning (ICML)*, 2015, 1–8.

[17] W. Kumwilaisak, T. Piriyatharawet, P. Lasang, and N. Thatphithakkul, "Image Denoising with Deep Convolutional Neural and Multi-directional Long Short-term Memory Networks Under Poisson Noise Environments", *IEEE ACCESS*, 8, 2020, 86998–7010.

[18] K. Papadimitriou and G. Potamianos, "End-to-end Convolutional Sequence Learning for ASL Fingerspelling Recognition", in *INTERSPEECH 2019*, 2019, 2315–9.

[19] M. Parelli, K. Papadimitriou, G. Potamianos, G. Pavlakos, and P. Maragos, "Exploiting 3d Hand Pose Estimation in Deep Learning-based Sign Language Recognition from RGB Videos", in *European Conference on Computer Vision*, 2020, 249–63.

[20] S. Pierre, E. David, Z. Xiang, M. Michael, F. Rob, and L. Yann, "OverFeat: Integrated Recognition, Localization and Detection Using Convolutional Networks", in *International Conference on Learning Representations (ICLR)*, 2014.

[21] N. Pugeault and B. R., "Spelling It Out: Real-time ASL Fingerspelling Recognition", in *2011 IEEE International Conference on Computer Vision Workshop (ICCV Workshops)*, 2011, 1114–9.

[22] A. Rakowski and L. Wandzik, "Hand Shape Recognition Using Very Deep Convolutional Neural Networks", in *2018 International Conference on Control and Computer Vision (ICCCV)*, 2018, 8–12.

[23] L. Rioux-Maldague and P. Giguère, "Sign Language Fingerspelling Classification from Depth and Color Images Using a Deep Belief Network", in *2014 Canadian Conference on Computer and Robot Vision (CRV)*, 2014, 92–7.

[24] K. Rodríguez Otiniano and G. Camála-Chávez, "Finger Spelling Recognition from RGB-D Information Using Kernel Descriptor", in *2013 XXVI Conference on Graphics, Patterns and Images*, 2013, 1–7.

[25] B. Shi and K. Livescu, "Multitask Training with Unlabeled Data for End-to-end Sign Language Fingerspelling Recognition", in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, 389–96.

[26] B. Shi, A. M. D. Rio, J. Keane, D. Brentari, G. Shakhnarovich, and K. Livescu, "Fingerspelling Recognition in the Wild with Iterative Visual Attention", in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, 5399–408.

[27] B. Shi, A. M. D. Rio, J. Keane, J. Michaux, D. Brentari, G. Shakhnarovich, and L. K., "American Sign Language Fingerspelling Recognition in the Wild", in *2018 IEEE Spoken Language Technology Workshop (SLT Workshop)*, 2018, 145–52.

[28] S. Yang, W. Chen, W. Huang, and Y. Chen, "DDaNet: Dual-path Depth-aware Attention Network for Fingerspelling Recognition Using RGB-D Images", *IEEE Access*, 9, 2021, 7306–22.

[29] L. Yann and B. Yoshua, "Convolutional Networks for Images, Speech, and Time Series", in *The Handbook of Brain Theory and Neural Networks*, 1995.

[30] L. Yujian and L. Bo, "A Normalized Levenshtein Distance Metric", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017, 1091–5.

[31] C. Zhang, X. Yang, and T. Y., "Histogram of 3D Facets: A Characteristic Descriptor for Hand Gesture Recognition", in *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2013, 1–8.

[32] H. Zhuang, M. Yang, Z. Cui, and Q. Zheng, "A Method for Static Hand Gesture Recognition Based on Non-Negative Matrix Factorization and Compressive Sensing", *International Journal of Computer Science*, 44, 2017, 52–9.