INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

End-to-end Convolutional Sequence Learning for ASL Fingerspelling Recognition

Katerina Papadimitriou, Gerasimos Potamianos

Electrical and Computer Engineering Department, University of Thessaly, Volos 38221, Greece
aipapadimitriou@uth.gr, gpotam@ieee.org

Abstract

Although fingerspelling is an often overlooked component of sign languages, it has great practical value in the communication of important context words that lack dedicated signs. In this paper we consider the problem of fingerspelling recognition in videos, introducing an end-to-end lexicon-free model that consists of a deep auto-encoder image feature learner followed by an attention-based encoder-decoder for prediction. The feature extractor is a vanilla auto-encoder variant, employing a quadratic activation function. The learned features are subsequently fed into the attention-based encoder-decoder. The latter deviates from traditional recurrent neural network architectures, being a fully convolutional attention-based encoder-decoder that is equipped with a multi-step attention mechanism relying on a quadratic alignment function and gated linear units over the convolution output. The introduced model is evaluated on the TTIC/UChicago fingerspelling video dataset, where it outperforms previous approaches in letter accuracy under all three, signer-dependent, -adapted, and -independent, experimental paradigms.

Index Terms: fingerspelling recognition, attention-based CNN encoder-decoder, auto-encoder feature extractor

1. Introduction

Automatic sign language (SL) recognition from video constitutes a popular field of research that has attracted increasing interest over the last three decades. Considering SL recognition as a linguistic task, many researchers have introduced automatic speech recognition-like approaches, with signs being treated analogously to phonemes or words. In this direction, several methods have been proposed, with the most commonly used ones being based on neural networks [1–5], fuzzy systems [6], and hidden Markov models [7–10].

However, among the proposed SL recognition schemes, limited effort has been devoted to the recognition of fingerspelling, which is commonly used for prominent words lacking in unique signs, such as names, technical terms, or foreign words. The majority of systems introduced for this purpose [11–14], though, suffer some restrictions concerning the limited vocabulary size and the scarcity of labeled data. In this paper, we focus on the recognition of American sign language (ASL) fingerspelling from videos, which corresponds to approximately 12-35% of ASL, and, regarding it as an image-to-text translation task, we seek to address the above issues by introducing a two-stage machine learning approach. For this purpose, we develop an end-to-end lexicon-free recognition method, consisting of two main components that are trained jointly: a deep auto-encoder image feature extractor and an attention-based encoder-decoder for the prediction task. Specifically, we assume that each image sequence constituting a letter-spelled word is fed into a feature extractor consisting of a variation of the so-called vanilla auto-encoder [15], complemented with a quadratic activation function. Subsequently, the features are passed through an attention-based encoder-decoder scheme, generating the predicted letter sequence. In speech recognition and machine translation, the most dominant models map the input sequence to the output using recurrent neural network (RNN) encoder-decoders [16, 17]. In this paper, we introduce a fully convolutional attention-based encoder-decoder, similar in spirit to the model of [18], using a multi-step attention mechanism relying on gated linear units [19] over the convolution output. Nevertheless, there are various key differences between [18] and our model. Firstly, the proposed model is complemented with an introduced quadratic alignment score function. Additionally, our model deviates from using input element position embeddings in the encoder-decoder inputs, incorporating instead previous alignment decisions in the current alignment score calculation through an input feeding scheme.

More precisely, the main contributions of this work lie in: (i) the introduced quadratic function for calculating the activations of the auto-encoder module; (ii) the design of a novel fully convolutional attention-based encoder-decoder architecture complemented with an input feeding scheme applied to the fingerspelling recognition task; and (iii) the proposed quadratic alignment function for computing the annotation scores.

We evaluate our approach on the TTIC/UChicago ASL fingerspelling video dataset (Chicago-FSVid), containing $2.4k$ word instances expressed by 4 signers [20]. The best prior results on the open-vocabulary fingerspelling recognition task are reported in [20], based on conditional random fields and a deep neural network feature learner, as well as in [21], which is similar in spirit to the proposed model, but differing in the activation function of the vanilla auto-encoder and the RNN attention-based encoder-decoder. We compare our approach experimentally against three alternative systems examining the effect of our architecture on the ASL fingerspelling recognition task, and we achieve letter accuracy improvements of 5.1%, 4.1%, and 16.2% under the signer-dependent, -adapted and -independent experimental paradigms, respectively.

2. Methods

Sign language fingerspelling recognition from videos can be treated as a sequence-to-sequence prediction problem addressed by an encoder-decoder approach. A source word $x = (x_1, x_2, ..., x_m)$, represented by m raw image frames, is processed producing a sequence of image features $z = (z_1, z_2, ..., z_m)$, which in turn passes through an encoder-decoder module generating the predicted letter sequence $y = (y_1, y_2, .., y_n)$. The general architecture of the proposed model consists of two main components, which are trained jointly (they can also be trained separately): an image feature extractor and an attention-based encoder-decoder for the letter prediction

http://dx.doi.org/10.21437/Interspeech.2019-2422

task. Like many recent sequence prediction models [16, 22], the attention-based model generates y that derives from z.

2.1. Feature extractor

2.1.1. Vanilla auto-encoder with ReLU (VAE1)

The vanilla auto-encoder [15] is a neural network that maps the input image $x \in \mathbb{R}^{d_x}$ to a latent variable $z \in \mathbb{R}^{d_z}$ through an encoder, with $d_z < d_x$, and then a decoder reconstructs the input $\tilde{x} \in \mathbb{R}^{d_x}$ based on z. The VAE1, which uses multi-layer perceptron (MLP) encoding-decoding and a ReLU activation function, aims to minimize the error function $\mathcal{L}(x) = \|x - \tilde{x}\|^2$.

2.1.2. Vanilla auto-encoder with parabola (VAE2)

We introduce a variation of VAE1 that replaces the ReLU activation function with a parabolic function given by:

$$A(r) = (w^\top r + b)^2 \,, \tag{1}$$

with range $[0, \infty)$, r being the d-dimensional input vector, w representing the weight vector, and b denoting the bias value. To enhance the sparsity and the efficiency of the activations, we perform a softmax function for normalizing the outputs to $[0, 1]$.

2.1.3. Histogram of oriented gradient (HOG)

In contrast to the vanilla auto-encoder, the histogram of oriented gradient (HOG) [23] is a hand-engineered feature descriptor, which converts the input image x to the feature vector z. Here, the HOG feature vector of [20] is used.

2.2. Attention-based encoder-decoder

The latent variable sequence $z = (z_1, z_2, ..., z_m)$ derived from the feature extractor is fed to an encoder, generating state representations $h = (h_1, h_2, ..., h_m)$. Then, h is processed by the decoder, returning the elements of the output sequence $y = (y_1, y_2, ..., y_n)$, one by one. In attention-based architectures the context vector c_t, which denotes the likelihood of each chunk of image frames being relevant to the current output, is computed as the weighted sum of h_i at each time step t. Supposing that d_t denotes the hidden state of the decoder at time step t, the annotation scores a_{ti} are given by a score function normalized by softmax. To-date, several alternatives have been proposed for this task, like the alignment function in [16]:

$$a_{ti} = \text{softmax}(\tanh(W_h\, h_i + W_d\, d_t)) \,, \tag{2}$$

where W_h and W_d are the model parameters (weights and biases), and the one in [17]:

$$a_{ti} = \text{softmax}(h_i^\top d_t) \,. \tag{3}$$

Here, we introduce an alignment function, which includes a different scoring calculation, formulated as:

$$a_{ti} = \text{softmax}(W_h h_i + W_d\, d_t)^2 \,. \tag{4}$$

The context vector c_t, for $t = 1, 2, ..., N$, derives from the weighted sum of each encoder hidden state:

$$c_t = \sum_{i=1}^{m} (a_{ti}\, h_i) \,. \tag{5}$$

The predictive distribution of outputting letter y_t, $p(y_t | \, y_{<t}, z)$, is given by:

$$p(y_t | \, y_{<t}, z) = \text{softmax}(W_s\, \tilde{d}_t) \,, \tag{6}$$

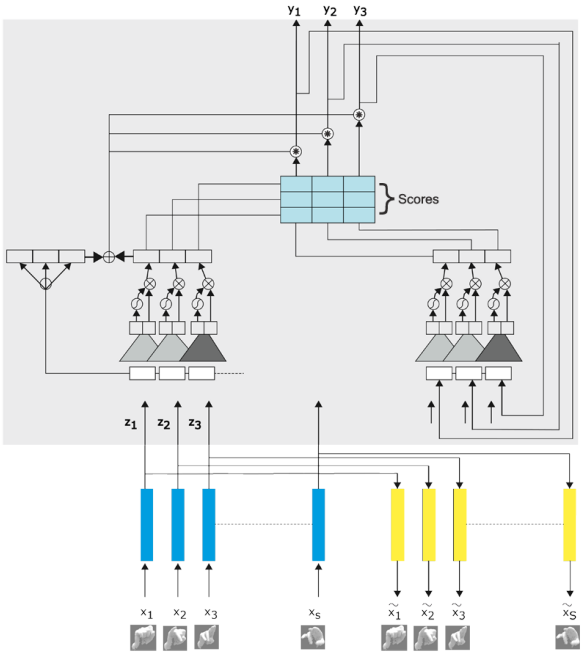

Figure 1: *General architecture of the proposed two-stage model with the gray region illustrating the attention-based convolutional encoder-decoder model fed by an auto-encoder module.*

where $\tilde{d}_t$ is the attentional hidden state derived from the concatenation of d_t and c_t as:

$$\tilde{d}_t = \tanh(W_c\, [c_t; d_t]) \,. \tag{7}$$

2.2.1. Attention-based recurrent encoder-decoder

Sequence-to-sequence prediction is mainly associated with RNN encoder-decoder models. A variety of attention-based RNN such schemes have been introduced, which differ in the RNN types and in the context vector calculation. The most popular RNN encoder-decoder alternatives are long short-term memory (LSTM) networks [21, 24] and gated recurrent units [25]. Recently, several architectures based on bi-directional encoders have been proposed [16, 26, 27]. Besides, various formulations do not conform to the way of computing context vector c, with [16] computing c_t by employing the previous decoder hidden state d_{t-1}, and [17] using the current decoder hidden state d_t to obtain the prediction.

2.2.2. Attention-based convolutional encoder-decoder

Inspired by [17, 18], we introduce an attention-based architecture relying on multi-layer convolutional encoder-decoder neural networks (see Fig. 1). Specifically, both encoder and decoder employ a convolutional block structure to compute the latent state representations h and d. For simplicity, we assume that the output of the l-th encoder's layer is expressed by $h^l = (h_1^l, h_2^l, ..., h_m^l)$ and the l-th layer of the decoder returns $d^l = (d_1^l, d_2^l, ..., d_n^l)$ hidden states. Considering a single-layer decoder with kernel width k, each output hidden state d_j^1 will be associated with k inputs. Thus, if we add multiple layers on top of each other, under the assumption that later layers will process k outputs of the previous ones, the resulting state will be related to more inputs than previously. More precisely, stacking 5 layers with kernel width $k = 5$, leads each hidden state d_j^5 output to be determined by 21 inputs. Thus, k concatenated input

Table 1: *Letter error rates (%) of various fingerspelling systems, evaluated on the Chicago-FSVid corpus test set under a signer-dependent (SD), signer-adapted (SA), and signer-independent (SI) experimental paradigm.*

Experimental paradigm		SD			SA			SI		
Encoder-Decoder	Features	HOG	VAE1	VAE2	HOG	VAE1	VAE2	HOG	VAE1	VAE2
RNN		11.1	8.1	7.3	29.1	23.5	25.8	50.3	43.7	35.3
CNN		10.2	8.0	5.9	27.8	20.5	21.6	48.7	35.8	32.5
Transformer		10.9	8.9	6.7	28.4	26.4	25.2	49.9	40.1	34.8
Proposed, alignment (2)		10.4	7.9	6.4	28.3	23.2	25.2	49.6	36.9	34.3
Proposed, alignment (3)		9.9	7.9	5.7	26.8	20.9	21.6	48.3	33.5	31.9
Proposed, alignment (4)		9.7	7.2	**3.0**	26.4	20.8	**19.0**	44.9	30.7	**27.5**

elements embedded in D dimensions, $Z \in \mathbb{R}^{k \times D}$, are provided as inputs to a convolutional kernel $K \in \mathbb{R}^{2D \times kD}$, outputting $H \in \mathbb{R}^{2D}$, with twice the dimensionality of input elements.

As in [18], each layer composes of a one-dimensional convolution followed by a gated linear unit [19], which behaves as a gating scheme that assists in dealing with the convolution output H:

$$u(H) = A \otimes \sigma(B) \ , \tag{8}$$

with $u(H) \in \mathbb{R}^D$ denoting which of A inputs are relevant with the current target element through the sigmoid non-linear gate $\sigma(B)$, $A, B \in \mathbb{R}^D$ being the outputs of the convolution ($H = [AB] \in \mathbb{R}^{2D}$), and $\otimes$ indicating point-wise multiplication.

To smoothly optimize and leverage the performance of deep convolutional networks, residual functions with reference to each convolution input and layer output are added, as in [28]. Moreover, in order to retain the length of encoder convolutional layers output as the input length, we add zero padding to the inputs of each layer. On the other hand, for the decoder inputs, we add zero padding by $k-1$ elements on both sides and then remove k elements from the output end to avoid the inclusion of future information during decoding, as in [18, 29].

Compared to attention-based recurrent encoder-decoder architectures, a multi-layer CNN encoder-decoder model implies a multi-step attention mechanism [18, 30]. More precisely, as shown in Fig. 1, the provided latent variable z by the feature extractor is fed into the multi-layer convolutional encoder outputting the hidden states h_i^l for the top layer l. Consequently, the weights a_{ij}^l are computed through the alignment function (see Section 2.2) conferred upon the corresponding decoder layer d_j^l and each output of the last encoder layer h_i^l. The context vectors c_j^l, which derive from the weighted sum of each output of the last encoder layer combined with the embeddings e_i of the input elements z_i in distributional space:

$$c_j^l = \sum_{i=1}^{m} a_{ij}^l (h_i^l + e_i) \ , \tag{9}$$

are in turn fed to the next decoder layer exploiting specific information during attention calculation. Once context vector c_j^l of the last layer has been computed, it is concatenated with the decoder hidden state returning the attentional vector $\tilde{d}_t$.

In contrast to [18], we avoid the addition of input elements z_i position embeddings in the encoder-decoder inputs, since it seems to hurt performance. Instead, we use an input feeding scheme [17], where the attentional vectors $\tilde{d}_t^l$ are concatenated with inputs at the following time step. Input feeding turns out beneficial to our model, owing to the fact that previous attentional vectors are taken into account during current alignment scores calculation. Thus, through this modification, our model turns into a fully connected deep network in both directions (vertical and horizontal) that deploys substantially previous alignment information during the estimation of new ones.

3. Experiments

3.1. Datasets and experimental framework

We conduct experiments on the TTIC/UChicago ASL fingerspelling video database (Chicago-FSVid) [20, 21]. This contains 2 repetitions of 300 word instances expressed by 4 native ASL signers, namely 600 fingerspelled sequences per signer, comprising of names, English and foreign words. Specifically, the corpus consists of 347,962 hand region frames generated by the preprocessing procedure in [20], including hand detection and segmentation. Additionally, HOG features for each frame are available. As in [21], we utilize supplementary hand bounding boxes acquired by the dataset of [31], which includes 65,774 ASL hand gesture frames, as well as the dataset of [32] with 63,175 handshapes, enhancing mostly signer-independent recognition performance. These frames, which consist of external handshapes obtained by different signers in various backgrounds, are employed in pretraining the auto-encoder, improving feature learning ability. To reduce processing time, each frame fed to the network is previously scaled to 64×64 pixels. We evaluate our models under a signer-dependent (SD), signer-independent (SI), as well as a signer-adapted (SA) experimental paradigm, using the same setup as in [20].

3.2. Evaluated systems

HOG/VAE1/VAE2 + RNN enc-dec: The image features, acquired by any of the three feature extractors, are directly fed to an attention-based RNN encoder-decoder, similar to the network in [21], using an LSTM. The model consists of a one-layer encoder and a one-layer decoder, both with 128 hidden units and 128-dimensional letter embeddings.

HOG/VAE1/VAE2 + CNN enc-dec: The respective image features are used as inputs to an attentional CNN encoder-decoder network with the architecture of [18]. This model comprises of a 5-layer encoder and a 5-layer decoder, both with 100 hidden units and kernel width 5.

HOG/VAE1/VAE2 + Proposed enc-dec: All image features are extracted and passed through the proposed attention-based CNN encoder-decoder. For fair comparisons, we use the same setup as in the CNN encoder-decoder above.

HOG/VAE1/VAE2 + Transformer enc-dec: The features are fed to the transformer [33], which is a multi-head attention-based architecture without the inclusion of an RNN or CNN. We use a 6-layer transformer with 8 heads for transformer self-attention and 1024-dimension hidden transformer feed-forward.

3.3. Implementation details

The auto-encoder is first trained using the external image frames through the auto-encoder loss function alone (see Sec-

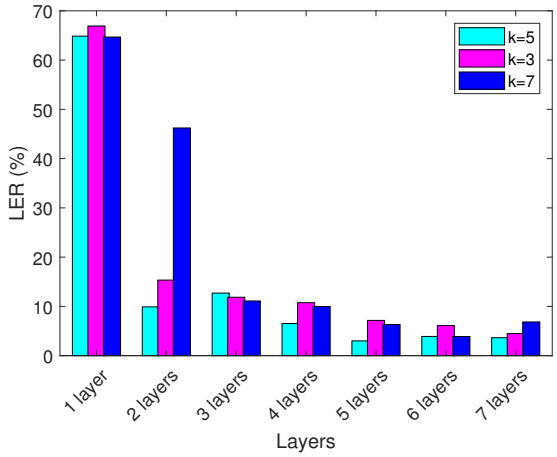

Figure 2: *LER (%) of the proposed model with different number of layers and kernel widths in the SD case.*

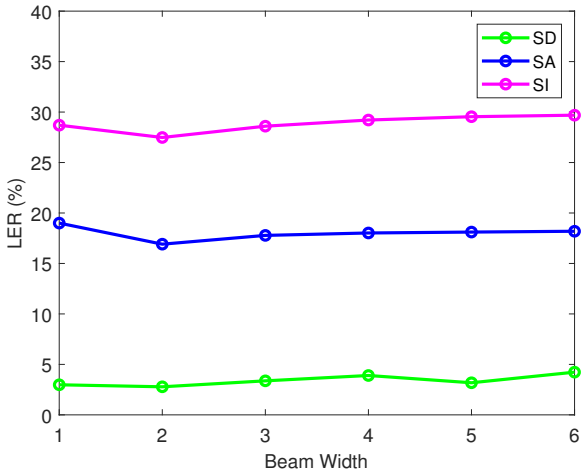

Figure 3: *LER (%) of the proposed model with different beam widths under the SD, SA, and SI experimental paradigms.*

tion 2.1.1). Subsequently, by employing the pretrained auto-encoder, the whole model is trained using the multitask loss function mentioned in [21]. We use a stacked auto-encoder with a 2-layer MLP encoder and a 2-layer MLP decoder of dimension 100. To initialize weights, Xavier initialization [34] is conducted. For auto-encoder network training, stochastic gradient descent with momentum is employed with an initial learning rate of 0.004, decreasing by a factor of 0.8. A mini-batch size of 64 images is used, and the resulting latent representation z is a 100-dimensional vector.

For the encoder-decoder scheme, the size of hidden states is fixed at 100, and letter embeddings with a dimensionality of 100 are used. Network training employs the Adam optimizer [35] with an initial learning rate of 0.001, decaying by a factor of 0.8. To obtain a better matching of a target element, during decoding the beam search strategy of [36] is applied, with beam width equal to 2. The implementation is conducted in PyTorch, and training is carried out on a Nvidia GTX 1050 Ti GPU.

3.4. Performance evaluation

The proposed model turns out superior to the considered alternatives in terms of the average letter error rate (LER), %. As demonstrated in Table 1, the proposed approach, using VAE2 and the introduced alignment function (4), outperforms the other models in all three experimental paradigms (SD, SA, and SI), improving over the best reported results of [20,21] by 5.1% under SD training/testing, 4.1% in the SA case, and 16.2% in the SI one. Additionally, the parabolic activation function in the vanilla auto-encoder yields consistent accuracy improvements under the three experimental paradigms over all models.

In Fig. 2, we depict the LER for different numbers of layers in the encoder-decoder and various kernel width sizes, under the SD experimental paradigm. As it may be observed, the best setup in the SD case is a 5-layer encoder-decoder with kernel width equal to 5, while the worst is a 1-layer encoder-decoder with kernel width 3. Clearly, deeper architectures benefit model performance. Additionally, in Fig. 3, we investigate the effect of beam search during prediction in all three experimental settings. It can be readily seen that, in all three cases, the best LER is achieved with beam width equal to 2. Notably, beam search width hardly affects model accuracy, especially under SD training/testing. Finally, in Fig. 4, we visualize the attention weights for all three settings using the three alignment functions, denoting with black color the best score. Note that we removed all duplicate entries from both ground-truth and predicted letter sequences. As it can be observed, the alignment between decoder output and image frames is generally monotonic, with the proposed model in the SD case providing the best one.

4. Conclusion

We presented an end-to-end lexicon-free model to address ASL fingerspelling recognition from video. Our model combines a vanilla auto-encoder variant for feature extraction and a fully convolutional attention-based encoder-decoder, which are trained jointly for letter sequence prediction. Compared to previous approaches, the proposed model yields letter accuracy improvements of 5.1%, 4.1%, and 16.2% under a signer-dependent, -adapted, and -independent experimental paradigm, respectively. In future work, we aim to evaluate our model on "in the wild" data, as well as to investigate the applicability of our approach to other sequence-to-sequence learning problems.

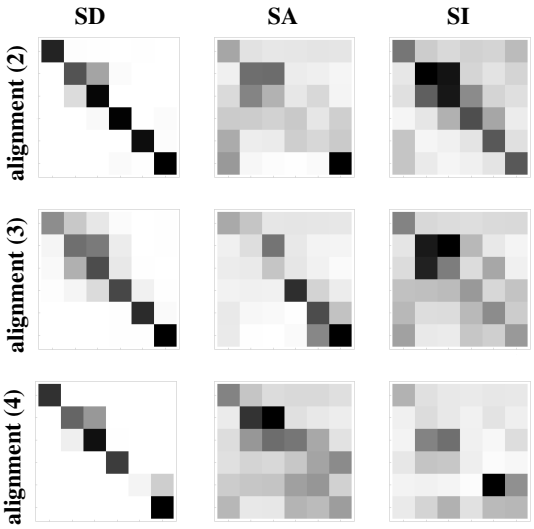

Figure 4: *Visualization of attention scores for word "effort", with the vertical axis depicting the source word and the horizontal the target word generated by the VAE2 feature extractor and the proposed encoder-decoder. All three experimental paradigms and all three alignment score functions are shown.*

5. References

[1] G. Anantha Rao, K. Syamala, P. V. V. Kishore, and A. S. C. S. Sastry, "Deep convolutional neural networks for sign language recognition," in *Proc. Conference on Signal Processing and Communication Engineering Systems (SPACES)*, 2018, pp. 194–197.

[2] P. Mekala, Y. Gao, J. Fan, and A. Davari, "Real-time sign language recognition based on neural network architecture," in *Proc. IEEE Southeastern Symposium on System Theory*, 2011, pp. 195–199.

[3] V. Bheda and D. Radpour, "Using deep convolutional networks for gesture recognition in American Sign Language," *Computing Research Repository*, 2017, arXiv:abs/1710.06836v3.

[4] J. Huang, W. Zhou, H. Li, and W. Li, "Sign Language recognition using 3D convolutional neural networks," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, 2015.

[5] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen, "Sign language recognition using convolutional neural networks," in *Proc. European Conference on Computer Vision Workshops (ECCVW)*, vol. LNCS 8925, 2015, pp. 572–578.

[6] M. Sarfraz, Y. A. Syed, and M. Zeeshan, "A system for sign language recognition using fuzzy object similarity tracking," in *Proc. International Conference on Information Visualisation (IV)*, 2005, pp. 233–238.

[7] T. Starner, J. Weaver, and A. Pentland, "Real-time American sign language recognition using desk and wearable computer based video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1371–1375, 1998.

[8] C. Vogler and D. Metaxas, "Parallel hidden Markov models for American sign language recognition," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, vol. 1, 1999, pp. 116–122.

[9] K. Grobel and M. Assan, "Isolated sign language recognition using hidden Markov models," in *Proc. IEEE International Conference on Systems, Man, and Cybernetics (ICSMC)*, vol. 1, 1997, pp. 162–167.

[10] P. Dreuw, D. Rybach, T. Deselaers, M. Zahedi, and H. Ney, "Speech recognition techniques for a sign language recognition system," in *Proc. Interspeech*, 2007, pp. 2513–2516.

[11] B. Shi, A. Martinez Del Rio, J. Keane, J. Michaux, D. Brentari, G. Shakhnarovich, and K. Livescu, "American Sign Language fingerspelling recognition in the wild," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 145–152.

[12] P. Goh and E. Holden, "Dynamic fingerspelling recognition using geometric and motion features," in *Proc. International Conference on Image Processing (ICIP)*, 2006, pp. 2741–2744.

[13] S. Liwicki and M. Everingham, "Automatic recognition of fingerspelled words in British Sign Language," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2009, pp. 50–57.

[14] S. Ricco and C. Tomasi, "Fingerspelling recognition through classification of letter-to-letter transitions," in *Proc. Asian Conference on Computer Vision (ACCV)*, vol. LNCS 5996, 2010, pp. 214–225.

[15] P. Baldi, "Autoencoders, unsupervised learning and deep architectures," in *Proc. ICML Workshop on Unsupervised and Transfer Learning (UTLW)*, vol. PMLR 27, 2012, pp. 37–49.

[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *Computing Research Repository*, 2014, arXiv:abs/1409.0473v7.

[17] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015, pp. 1412–1421.

[18] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *Computing Research Repository*, 2017, arXiv:abs/1705.03122v3.

[19] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. International Conference on Machine Learning (ICML)*, 2017, pp. 933–941.

[20] 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*, vol. 46, pp. 209–232, 2017.

[21] B. Shi and K. Livescu, "Multitask training with unlabeled data for end-to-end sign language fingerspelling recognition," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, pp. 389–396.

[22] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4960–4964.

[23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2005, pp. 886–893.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computing*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[26] J. Zhou, Y. Cao, X. Wang, P. Li, and W. Xu, "Deep recurrent models with fast-forward connections for neural machine translation," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 371–383, 2016.

[27] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *Computing Research Repository*, 2016, arXiv:abs/1609.08144v2.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[29] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proc. International Conference on Machine Learning (ICML)*, 2016, pp. 1747–1756.

[30] S. Chollampatt and H. T. Ng, "A multilayer convolutional encoder-decoder neural network for grammatical error correction," in *Proc. AAAI Conference on Artificial Intelligence*, 2018.

[31] N. Pugeault and R. Bowden, "Spelling it out: Real-time ASL fingerspelling recognition," in *Proc. IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2011, pp. 1114–1119.

[32] T.-K. Kim and R. Cipolla, "Canonical correlation analysis of video volume tensors for action categorization and detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 8, pp. 1415–1428, 2009.

[33] 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 (NIPS)*, vol. 30, 2017, pp. 5998–6008.

[34] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. International Conference on Artificial Intelligence and Statistics*, vol. PMLR 9, 2010, pp. 249–256.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computing Research Repository*, 2014, arXiv:abs/1412.6980v9.

[36] M. Freitag and Y. Al-Onaizan, "Beam search strategies for neural machine translation," in *Proc. Workshop on Neural Machine Translation*, 2017, pp. 56–60.