Non-autoregressive Modeling for Sign-gloss to Texts Translation

Fan Zhou, Tim Van de Cruys
Center for Computational Linguistics, KU Leuven
{fan.zhou, tim.vandecruys}@kuleuven.be

Abstract

Automatic sign language translation has seen significant advancements, driven by progress in computer vision and natural language processing. While end to end sign-to-text translation systems are available, many systems still rely on a gloss-based representation–an intermediate symbolic representation that functions as a bridge between sign language and its written counterpart. This paper focuses on the gloss-to-text (gloss2text) task, a key step in the sign-to-text translation pipeline, which has traditionally been addressed using autoregressive (AR) modeling approaches. In this study, we propose the use of non-autoregressive (NAR) modeling techniques, including non-autoregressive Transformer (NAT) and diffusion models, tailored to the unique characteristics of gloss2text. Specifically, we introduce PointerLevT, a novel NAT-based model designed to enhance performance in this task. Our experiments demonstrate that NAR models achieve higher accuracy than pre-trained AR models with less data, while also matching the performance of fine-tuned AR models such as mBART. Furthermore, we evaluate inference speed and find that NAR models benefit from parallel generation, resulting in faster inference. However, they require more time to achieve an optimal balance between accuracy and speed, particularly in the multistep denoising process of diffusion models. All our code is publicly available at https://github.com/louisefz/non-autoregressive_signlang

1 Introduction

Deafness and hearing loss affect over 1.5 billion people worldwide, with 430 million experiencing disabling hearing loss[1]. Sign languages serve as an alternative to verbal speech, yet communication barriers between deaf individuals and non-sign language users can lead to social isolation and limited access to essential services. To mitigate these challenges, researchers have developed sign language translation systems such as WeCapable[2] and Hand Talk[3]. However, most existing systems focus on recognizing individual signs rather than capturing the full grammatical complexity of sign languages (Tolba and Elons, 2013; Masood et al., 2018; Rastgoo et al., 2021).

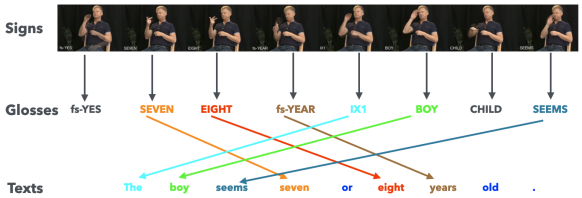

Figure 1: Framed sign video (sourced from (Börstell, 2022)) is converted into glosses, and then to texts.

A promising alternative is sign glosses, a written representation of sign language that captures the core meaning of signs[4]. Unlike standard written texts, glosses follow distinct linguistic rules in grammar, word selection, and sequential expression. Converting sign glosses into natural text—known as the gloss-to-text (gloss2text) problem (see Figure 1)—is typically framed as a low-resource machine translation task. Due to the scarcity of parallel gloss-text data, traditional approaches often rely on data augmentation and AR training to adapt neural models for this task (Camgoz et al., 2018; Yin and Read, 2020).

AR models are widely used in NLP and have demonstrated strong performance in numerous tasks (Gillioz et al., 2020; Black et al., 2022; Bevilacqua et al., 2022). However, they come with inherent limitations, including strong dependence

[1] https://www.who.int/health-topics/hearing-loss#tab=tab_1

[2] https://wecapable.com/tools/text-to-sign-language-converter/

[3] https://apps.apple.com/us/app/hand-talk-asl-sign-language/id659816995

[4] https://www.lifeprint.com/asl101/topics/gloss.htm

Proceedings of Machine Translation Summit XX Volume 1, pages 220–230, June 23-27, 2025

on large datasets for effective learning, error accumulation during sequential generation, and high computational costs due to their step-by-step decoding nature. Given these constraints, and considering the unique characteristics of the gloss2text task—namely, its low-resource setting, high lexical overlap between glosses and text, and the challenge of inferring natural language from simplified gloss sequences—we explore NAR modeling as a promising alternative.

Our approach primarily focuses on improving effectiveness while also considering efficiency as a complementary aspect. To achieve this, we investigate two types of NAR models. The first approach is based on the edit-based Levenshtein Transformer (LevT) (Gu et al., 2019), which refines predictions iteratively through insertion and deletion operations. To enhance its performance, we introduce PointerLevT, an improved version that integrates a Pointer Network, allowing for better alignment between glosses and their corresponding textual representations. Edit-based NAR models are particularly well-suited for tasks requiring minimal corrections or modifications, balancing effectiveness and efficiency. The second approach leverages diffusion-based sequence models, including vanilla diffusion models (Ho et al., 2020) and DiffuSeq (Gong et al., 2022, 2023). These models condition their sampling steps on sign glosses, guiding the model through a probabilistic denoising process that refines a noisy sequence into the target text. Diffusion models offer robustness against noise, better handle variability in input, and provide stochasticity for possible exploration of text generation, which show potential to enhance gloss2text generation accuracy.

Our experiments show that these NAR models outperform AR counterparts trained from scratch on the same small dataset, demonstrating advantages in either effectiveness, efficiency, or both. This highlights the potential of NAR approaches for gloss2text task and similar low-resource and noisy-input NLP problems such as grammatical error correction, post-editing, and text infilling or modification.

2 Related Work

2.1 Gloss2Text

The gloss2text phase in the pipeline of sign-to-text translation is treated as a low-resource machine translation task (Camgoz et al., 2018; Yin and Read, 2020). Most existing studies have focused on enlarging available datasets and validating AR modeling approaches. To address the low-resource challenge, data augmentation techniques have been employed, including rule-based heuristics that exploit lexical similarities and syntactic variations between sign and spoken languages to generate artificial gloss-text pairs (Moryossef et al., 2021). To further mitigate resource limitations and enhance translation quality, ConSLT, a token-level contrastive learning framework, was proposed. It processes sign glosses through a Transformer model twice to generate positive pairs while treating tokens outside the sentence as negative pairs (Fu et al., 2022). Additionally, part-of-speech (POS) tags have been utilized to refine data augmentation strategies and improve the quality of generated samples (Liu et al., 2023).

2.2 Edit-based Non-autoregressive Transformer

NAR models offer fast inference but often struggle with generation quality. To address this issue, edit-based models have been developed to enhance effectiveness by refining generated outputs iteratively. Iteration-based NAR models were introduced to refine outputs. These models either use the previous iteration's results or a noisy version of the target sentence to initialize the decoder input (Lee et al., 2020). Insertion Transformer determines both the content to insert and its precise placement by leveraging concatenated slot representations (Stern et al., 2019). Levenshtein Transformer (LevT) employs a dual policy learning strategy during training and utilizes three distinct classifiers to determine the placement and quantity of token insertions, manage token deletions, and predict token content (Gu et al., 2019). ReorderNAT adopts a two-decoder approach. One decoder, equipped with cross-attention to the encoder, restructures the source sentence to align more closely with the target word order, thereby enabling more accurate word position decisions (Ran et al., 2021). Syntactic labels are incorporated as a form of supervision to enhance the learning process of discrete latent variables (Akoury et al., 2019). Bao et al. (Bao et al., 2022) developed a glancing sampling technique to effectively optimize latent variables.

2.3 Diffusion Models for Text Generation

Diffusion models, originally designed as latent variable models for continuous data, have been adapted

for text generation and are now recognized as a type of NAR model in the field of NLP (Li et al., 2023). These models typically operate through a multi-step process of sequential noising and denoising, gradually refining random noise into meaningful data samples (see Formulas 1 - 2). When applied to NAR text generation tasks, diffusion models iteratively refine intermediate outputs based on input data, offering a promising approach for handling complex control conditions and producing high-quality text (Li et al., 2022). Their ability to model intricate dependencies and generate coherent sequences through iterative denoising makes them a compelling alternative to traditional NAR approaches. Diffusion-LM incorporated an embedding layer into the diffusion model to turn discrete tokens into a continuous form, to be able to adapt diffusion's attribute (Li et al., 2022). DiffuSeq introduced a partial noising strategy to integrate conditional text with the continuous diffusion process (Gong et al., 2022, 2023).

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad (1)$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t)) \quad (2)$$

The loss function in diffusion models typically minimizes the difference between the predicted noise and the actual noise added during the forward process. Evidence Lower Bound (ELBO) is used to represent the divergence between the forward and backward processes in the diffusion model (see Formula 3). In diffusion models, it is typically assumed that $\Sigma_\theta(x_t, t)$ is fixed (e.g., $\Sigma_\theta = \beta_t I$), so the KL divergence simplifies to the difference in means (MSE) (see Formula 4).

$$L_{\text{ELBO}} = \mathbb{E}_q \left[\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right] \quad (3)$$

$$D_{\text{KL}}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)) \propto \|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2 \quad (4)$$

3 LevT and PointerLevT for Gloss2Text

Considering the words shared between glosses and texts, the Levenshtein Transformer (LevT) (Gu et al., 2019) is chosen to fit our task. In addition, we also propose PointerLevT with improved performance.

3.1 Levenshtein Transformer

The Levenshtein Transformer (LevT) follows an encoder-decoder architecture. Similar to a vanilla Transformer, LevT's encoder processes the input sequence through layers of self-attention and feed-forward networks, creating a set of representations that encapsulate the contextual information of the input. The decoder of LevT, operating in a NAR mode, uses these encoded representations (*H*) together with input (*H'*) to generate outputs.

In the training process, its decoder simultaneously passes hidden states into the three classifiers, and the training objective for LevT includes deletion loss, insertion loss, and placeholder insertion loss (see Formula 5).

During inference, the three operations are applied sequentially in each iteration: first deleting tokens, then inserting placeholders, and finally replacing placeholders with new tokens. The outputs from the previous iteration serves as the input of the next iteration in the decoder during the inference stage, and iterations continue until accurate output is generated.

$$\begin{aligned}\mathcal{L}(\theta) = \mathbb{E}_{y_{\text{del}} \sim d_{\text{del}}} \left[\sum_{d_k^* \in d_*} \log \pi_\theta^{\text{del}}(d_k^* | i, y_{\text{del}}) \right] \\ + \mathbb{E}_{y_{\text{ins}} \sim d_{\text{ins}}} \left[\sum_{p_i^* \in P^*} \log \pi_\theta^{\text{plh}}(p_i^* | i, y_{\text{ins}}) + \right. \\ \left. \sum_{t_i^* \in t_*} \log \pi_\theta^{\text{tok}}(t_i^* | i, y_{\text{ins}}) \right] \quad (5)\end{aligned}$$

3.2 PointerLevT

LevT reorders the sequence through editing operations in the inference stage, with time complexity of $O(n \times m)$. To reduce the number of edit operations and accelerate the inference time in the decoder part, we propose PointerLevT to incorporate a reordering method in the encoder part, viz. the pointer network (Vinyals et al., 2015). Theoretically, the time complexity can be reduced to $O(n \log m)$. In the context of the gloss2text task, pointer neural networks help solve position problems and are expected to reduce the number of edit operations in the LevT decoder.

This involves putting the traditional inter-attention between the decoder's query and the encoder's key with intra-attention within the encoder itself, simplifying the model architecture, and replacing Bahdanau attention (Bahdanau et al., 2014) with self-attention from vanilla Transformer

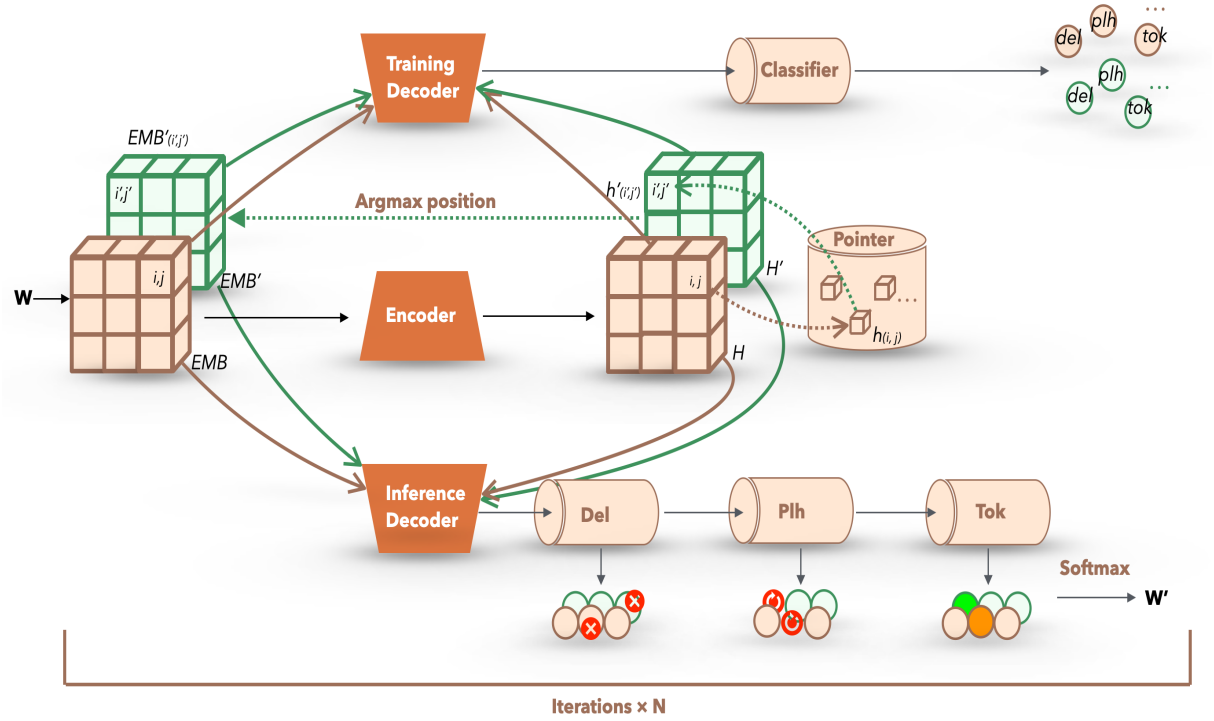

Figure 2: Overview of the LevT architecture (highlighted in red), illustrating its encoder and the decoding process of its decoder during both training and inference stages (Gu et al., 2019). The architecture of PointerLevT (highlighted in green) is also presented. The pointer network reorders the positions of hidden states, and these reordered positions are used to form the reordered word embeddings. The training and inference decoders are shared between LevT and PointerLevT, with both decoders taking as input of both the word embeddings (*EMB* or *EMB'*) and the encoder's final hidden states (*H* or *H'*).

(Vaswani et al., 2017). This change aims to reduce the complexity of the model architecture. Once the attention scores are obtained, the reordered encoder output is generated by the multiplication between the attention scores and encoder outputs (value). To determine the reordered source sequences, argmax is applied to the attention scores to get the reordered positions of the source sequence, which can be seen as the encoder's prediction. To ensure there is no duplication of predicted argmax positions, we use the Sinkhorn layer (Mena et al., 2018) which are differentiable modules inspired by the Sinkhorn algorithm (Sinkhorn and Knopp, 1967) and alternately rescales all rows and all columns of the matrix to sum to 1 as the normalization function (see Appendix A). In this setup, the reordering loss is calculated between the predicted reordered source sequences and the gold standard sequences using the cross-entropy function. There are pseudo codes for the PointerLevT modeling (see Figures 2-3 for details).

In terms of the loss function for this architecture, in addition to the three types of loss in the original LevT (deletion, insertion, and placeholder), there is an additional loss component for reordering. This reordering loss is calculated between the correctly ordered sentences and the predicted reordered input of the encoder (see Formula 6).

$$L_{\text{PointerLevT}} = \alpha \dot{L}_{\text{CE}}(y_{\text{true}}, y_{\text{reorder\_pred}}) + L_{\text{LevT}} \quad (6)$$

4 Diffusion Modeling for Gloss2Text

The motivation for using diffusion models in the gloss2text problem lies in the nature of sign glosses, which are written representations of sign gestures and lack the syntactic and semantic richness of standard texts. This makes them comparable to partially noised texts, creating an opportunity for models to explore denoising pathways for original sentence recovery. In this context, glosses serve as conditions that guide the denoising process. Guided by this intuition, we employ diffusion models, which gradually generate complex text distributions from standard Gaussian noise, effectively capturing the diversity and uncertainty between glosses and texts for potentially improved mapping.

Algorithm 1 Levenshtein Transformer with Pointer Network for Reordering

Input: Input sequence X, Target sequence Y, Encoder layers L_{enc}, Decoder layers L_{dec}, Vocabulary size V
Output: Output sequence by Levenshtein Transformer's decoder
Initialization:
Initialize encoder with L_{enc} layers and vocabulary size V
Initialize decoder with L_{dec} layers and vocabulary size V
Initialize pointer network
function ENCODE(X)
    Encoded representation $H \leftarrow$ Encoder(X)
    return H
end function
function POINTERNETWORK(X,H)
    Reordered Position $P \leftarrow$ argmax(Sinkhorn(Self-attention(H)))
    Reordered Encoded representation $H' \leftarrow$ H $\cdot$ Self-attention(H)
    Reordered Input $X' \leftarrow P(X)$
    return H', X'
end function
function DECODE(H', Y_{prev})
    if iteration == 0 **then**
        $Y_{prev} \leftarrow$ PointerNetwork.Reorder(X')
    end if
    while ITERATION < MAX **do**
        Decoded Output $Y_{pred} \leftarrow$ Decoder(H', Y_{prev})
        if $Y_{pred} == Y$ **then**
            break
        end if
    end while
    return Y_{pred}
end function
Training:
for each epoch in epochs **do**
    for each $(X, Y_{reordered}, Y_{good})$ in dataset **do**
        $H \leftarrow$ Encode(X)
        $X' \leftarrow$ Initialize with start token
        $Y_{pred} \leftarrow$ Decode(H', X')
        Calculate loss: Loss $\leftarrow \alpha \cdot$ CrossEntropy($X', Y_{reordered}$) + LevT three edit losses)
        Backpropagate loss and update model parameters
    end for
end for

Figure 3: Pseudocode of PointerLevT process

In diffusion model, sign glosses serve as conditions to guide the denoising process for controllable generation. Different from label-based controllable generation by diffusion models which reply on distinct labels such as classifier guidance (Li et al., 2022) and classifier-free guidance (Ho and Salimans, 2022) diffusion models, Seq2Seq generation is condition on source sequences, not on labels. In order to produce a target sequence $\mathbf{w_y}$ conditioned on the source sequence, we use DiffuSeq's method (Gong et al., 2022) (see Figure 4) to concatenate source $\mathbf{x_t}$ with target $\mathbf{y_t}$ (see Formula 7), and only partially noise targe sequences and denoise the target with the unnoised source. Word embedding from an existing pre-trained language model is used to convert discrete tokens into embeddings for diffusion continuous attributes at the begining of the forward process, and to turn the continuous representation to tokens at the end of the denoising process.

$$\mathbf{z}_t = \mathbf{x}_t \oplus \mathbf{y}_t, \quad \text{for } t \in [0, T] \tag{7}$$

The training loss function (Formula 8) is based on the variational latent boundary (VLB), aiming to optimize the variational lower bound on the objective function. In this case, the objective function simplifies the KL divergence into end-to-end MSE (Formulas 9 - 11, see Appendix B for more details).

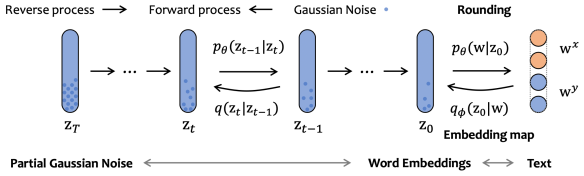

Figure 4: DiffuSeq noising and denoising processes with embedding and rounding through embedding layer, sourced from (Gong et al., 2022)

$$\mathcal{L}_{\text{simple}} \rightarrow \sum_{t=2}^{T} \|\mathbf{y}_0 - \tilde{f}_\theta(\mathbf{z}_t, t)\|^2 + \|\text{EMB}(\mathbf{w}^y) - \tilde{f}_\theta(\mathbf{z}_1, 1)\|^2 + \mathcal{R}(\|\mathbf{z}_0\|^2) \tag{8}$$

5 Experimental

5.1 Datasets

Our experiments are carried out using American sign language (ASL) datasets. The primary dataset used is the ASLG-PC12 corpus (Othman and Jemni, 2012), a substantial parallel corpus that aligns English written texts with ASL glosses. Given that certain experiments involve training models from scratch, the size of the ASLG-PC12 gloss-text parallel corpus is relatively small. To address this issue, data augmentation techniques are employed to generate additional artificial datasets. Artificial glosses are generated from corresponding standard texts from Wikipedia[5], leveraging the features of sign glosses based on sign linguistics. Linguistic features of sign glosses are analyzed from differences between sign language glosses and spoken language, which includes the lack of word inflection, the omission of punctuation and individual words, and syntactic diversity. Consequently, the corresponding heuristics for generating pseudo-glosses from spoken language involve the lemmatization of spoken words, POS-dependent and random word deletion, and random word permutation (Moryossef et al., 2021). This research follows these rules for data augmentation, ensuring the creation of robust pseudo-gloss datasets (See Table 1 for more dataset details).

The overall datasets are divided into train, validation and test datasets with proportion of 70%,

[5] https://www.kaggle.com/datasets/mikeortman/wikipedia-sentences/data

Datasets	Authentic	Artificial
# sentences	87,710	87,710
# words	1,151,110	1,687,804
# glosses	1,029,995	1,079,168
word vocab	22,070	120,273
gloss vocab	16,120	75,069
avg sentence len	13.124	19.243
avg gloss len	11.743	12.304
Train	61,397	61,397
Validation	13,157	13,156
Test	13,157	13,156

Table 1: Dataset used in experiments

15% and 15%. This means that our test dataset is a mixture of both artificial (through data augmentation) and real parallel data. To check how our model performs on real data, we create two test datasets. The first batch is the real dataset in which the parallel data is true glosses with corresponding true standard text translation (**Real** test data); while the second one is all the test data, the mixture of both authentic and artificial test data (**All** test data).

5.2 Evaluation Metrics

To evaluate the performance of different architectures, automatic metrics are used to assess both effectiveness and efficiency. Effectiveness metrics measure how well the predicted outputs align with reference texts, while efficiency metrics focus on the computational cost of model training and inference.

Effectiveness metrics include several measures to assess the quality of generated outputs. **Length** comparison helps determine whether the predictions are appropriately sized relative to the reference texts, highlighting potential tendencies toward overly brief or excessively long outputs. **Accuracy** measures whether the model correctly generates words and tokens, with **token-level accuracy** offering a finer-grained analysis to detect formatting errors. **Levenshtein distance** (Levenshtein et al., 1966) evaluates the syntactical alignment of predicted sequences by counting the minimum number of insertions, deletions, and substitutions required to match reference texts. Additionally, the **BLEU score** (Papineni et al., 2002) assesses the fluency and adequacy of generated sequences by comparing n-gram overlaps with reference texts. We make use of the BLEU score implementation provided by the NLTK package[6].

Efficiency metrics focus on computational performance, specifically **training time** and **inference time**. Training time refers to the total duration required to optimize the model across multiple epochs, excluding validation time. Inference time measures the speed at which a trained model processes new inputs and generates outputs.

5.3 Model Setups

LevT and PointerLevT Both models are encoder-decoder models with 6 layers each, using multi-headed attention, layer normalization, dropout, and embeddings of size 512 from "facebook/mbart-large-cc25". Both models perform up to 10 decoding iterations during inference; they are each trained on 1 NVIDIA A100 GPU.

DiffuSeq DiffuSeq incorporates source texts during training, converting both source and target texts into tensors using a pre-trained BERT tokenizer ("bert-base-uncased") with a vocabulary size of 30522 and $d_{\text{dimension}} = 128$. The noising process applies a square root scheduler $\beta_t = 1 - \sqrt{\frac{t}{T} + 0.0001}$ exclusively to target tensors, using an attention mask (0 for source, 1 for target) to distinguish them. Initially, noise is added uniformly to both tensors, but at the final noising timestep, source tensors are replaced with their original state, resulting in noised target tensors concatenated with unnoised source tensors. This concatenated state serves as the input for the denoising process, which treats both tensors uniformly. At the final denoising step, source tensors are reverted to their initial noising state. The process operates over 2000 timesteps. Multiple seeds are used to select the best outputs through Minimal Bayes Risk (MBR) decoding (Kumar and Byrne, 2004). The model is trained on 1 NVIDIA A100 GPU.

5.4 Baselines

Two types of autoregressive models serve as baseline models, including pre-trained language models, mBART (Chipman et al., 2022) and mT5 (Xue et al., 2020), as well as a small-scale mBART (of the same size as our non-autoregressive models) trained from scratch through knowledge distillation. Besides, a vanilla diffusion model also serves as a baseline for our condition-based diffusion model.

mBART and mT5 are multilingual encoder-decoder models designed for sequence generation

[6]https://www.nltk.org/api/nltk.translate.bleu_score.html

tasks. mBART ("mbart-large-cc25") uses 12 encoder and 12 decoder layers with 250,027 tokens, 1024 embedding size, and GELU activations. mT5 ("mt5-large") has 24 layers each for the encoder and decoder, 250,112 tokens, and similar GELU-based feed-forward layers with relative attention bias. mBART and mT5 are fine-tuned to adapt our dataset, serving as overall baseline models.

KD-mBART applies knowledge distillation using a smaller mBART student model (6 layers, 512 embeddings, 8 attention heads, and 2048 feed-forward dimensions) distilled from a fine-tuned mBART pre-trained teacher model.

The vanilla diffusion employs a 2000-step diffusion process with a square root scheduler for noise generation, defined by $\beta_t = 1 - \sqrt{\frac{t}{T} + 0.0001}$, where $T = 2000$. During denoising, timestep embeddings are integrated into a Transformer encoder and combined with position and input embeddings to incorporate temporal context. Instead of KL divergence, which can lead to instability and complexity, the model adopts MSE for more stable training. To prevent out-of-vocabulary issues, the vocabulary size is set to 13000, and custom word embeddings ($d_{\text{model}} = 128$) are jointly trained with the diffusion loss to optimize computational efficiency and control model size.

The details of models' used in the experiments are displayed in Table 3 in Appendix, including model's number of parameters, batch size of training and test, number of epochs or steps during training, the use of GPU during training and inference.

6 Results

6.1 Effectiveness Discussion

For the two edit-based models, the PointerLevT model demonstrates slightly better performance than the LevT model across most metrics. Specifically, PointerLevT achieves higher BLEU scores compared to LevT. Its word- and token-level accuracies are comparable to those of the fine-tuned mT5 model. Additionally, PointerLevT requires fewer edit operations, as indicated by a lower Levenshtein distance, particularly during inference on the real test set. This suggests that PointerLevT not only generates slightly more accurate sequences but also requires fewer modifications to match reference output. However, the performance difference between the two models remains marginal in general.

For diffusion model, DiffuSeq consistently outperforms vanilla Diffusion and other pre-trained models across multiple evaluation metrics. It achieves significantly higher word accuracy. Token-level accuracy follows a similar trend, highlighting DiffuSeq's superior ability to predict labels and individual tokens more accurately. DiffuSeq also demonstrates a lower Levenshtein Distance, indicating its outputs are closer to the reference texts and require fewer modifications. Additionally, DiffuSeq achieves a BLEU score, far exceeding vanilla Diffusion, The overall performance of effectiveness is comparable to that of the fine-tuned mBART, and also those of two edit-based NAT models.

Although two types of non-autoregressive models result in inferior performance on effectiveness compared to fine-tuned pre-trained models, they outperform their autoregressive counterparts trained from scratch through knowledge distillation. Examples of predicted output generated by each model are displayed in Table 4 in Appendix.

6.2 Efficiency Discussion

For edit-based NAR models, PointerLevT is significantly larger (5.29GB with over 550 million parameters) compared to LevT (1.95GB with 170 million parameters). PointerLevT requires 15.5 hours for training, whereas LevT takes 13.4 hours. However, PointerLevT achieves faster inference, completing the decoding process in 30 seconds, while LevT takes 42 seconds on the real test dataset. Both models were trained with a batch size of 16, but PointerLevT converged in just 10 epochs, whereas LevT required 12 epochs.

Diffusion models have a relatively small model size, with DiffuSeq at 363MB (91,225,274 parameters) and vanilla Diffusion at 334MB (87,336,792 parameters). However, conditioning significantly increases training time. DiffuSeq requires 150 hours to train, compared to just 44 hours for vanilla Diffusion. A similar trend is observed in inference. While DiffuSeq achieves the relatively short inference time per step (24.91s per step for full-text generation), it requires substantially longer—13.84 hours—to reach an optimal balance between accuracy and speed (see Figure 5). This suggests that while conditioning improves performance, it also reduces inference efficiency, which could be a limiting factor in real-time or resource-constrained applications where high effectiveness is needed.

Models	Effectiveness								Efficiency	
	Acc		Acc_token		LevD		BLEU		Train	Inference
	All	Real	All	Real	All	Real	All	Real		
Autoregressive models										
$mBART_{cc25}$	0.68*	0.69*	0.78*	0.82*	7.0*	3.81*	0.48*	0.61*	13.65h	805.25s
$mT5_{large}$	0.54	0.43	0.61	0.50	9.63	8.27	0.28	0.34	8.55h	637.53s
KD-mBART	0.11	0.16	0.38	0.55	14.73	11.36	0.01	0.01	13.75h	23.47s*
Non-autoregressive models										
LevT	0.37	0.37	0.52	0.53	12.08	8.22	0.07	0.13	13.4h	4.2s/42.11s
PointerLevT	0.37	0.41	0.52	0.54	11.11	6.26	0.07	0.15	15.5h	**3.1s/30.97s**
Vanilla Diffusion	0.14	0.17	0.17	0.20	17.51	14.97	0.01	0.01	44h	10.51s/5.84h
DiffuSeq	**0.58**	**0.77**	**0.67**	**0.80**	**11.05**	**3.53**	**0.21**	**0.47**	150h	24.91s/13.84h

Table 2: Models' effectiveness evaluation on all test datasets and real test datasets respectively; efficiency evaluation on real test datasets

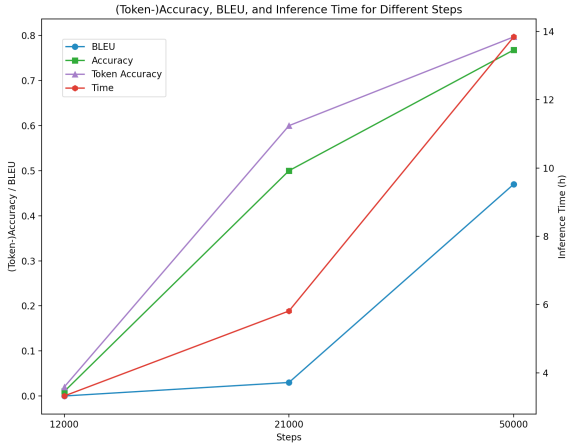

Figure 5: Generated sentences' accuracy, token-accuracy and BLEU in difference steps when DiffuSeq infers

7 Conclusion

This paper investigates two non-autoregressive (NAR) approaches to address the task of translating sign glosses into standard text (gloss2text). While large pretrained models such as mBART and mT5 typically achieve strong performance, our experiments reveal that these NAR models trained with a smaller size of dataset not only match the accuracy of fine-tuned large pre-trained language models but also significantly outperform smaller, distilled models of comparable capacity. In particular, the edit-based NAR strategy strikes a notably favorable balance between accuracy and efficiency, positioning it as a viable alternative to resource-intensive pretrained models. Meanwhile, the conditional diffusion-based approach attains very high accuracy and could benefit from further optimization to enhance its efficiency, making it well-suited for future research.

Moreover, the findings suggest that these methods can be applied to broader text-editing tasks resembling gloss2text, such as grammatical error correction and post-editing. This underscores the potential of novel modeling techniques in both sign language translation and general NLP applications.

Future work may focus on refining both edit-based and diffusion-based NAR models to bolster effectiveness and efficiency across diverse real-world scenarios.

Acknowledgments

We thank the anonymous reviewers for insightful feedback.

References

Nader Akoury, Kalpesh Krishna, and Mohit Iyyer. 2019. Syntactically supervised transformers for faster neural machine translation. *arXiv preprint arXiv:1906.02780*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yu Bao, Hao Zhou, Shujian Huang, Dongqi Wang, Lihua Qian, Xinyu Dai, Jiajun Chen, and Lei Li. 2022. latent-glat: Glancing at latent variables for parallel text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8398–8409.

Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings

as document identifiers. *Advances in Neural Information Processing Systems*, 35:31668–31683.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Carl Börstell. 2022. Introducing the signglossr package. In *Proceedings of the LREC2022 10th Workshop on the Representation and Processing of Sign Languages: Multilingual Sign Language Resources*, pages 16–23.

Necati Cihan Camgoz, Simon Hadfield, Oscar Koller, Hermann Ney, and Richard Bowden. 2018. Neural sign language translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7784–7793.

Hugh A Chipman, Edward I George, Robert E McCulloch, and Thomas S Shively. 2022. mbart: multidimensional monotone bart. *Bayesian Analysis*, 17(2):515–544.

Biao Fu, Peigen Ye, Liang Zhang, Pei Yu, Cong Hu, Yidong Chen, and Xiaodong Shi. 2022. Conslt: A token-level contrastive framework for sign language translation. *arXiv preprint arXiv:2204.04916*.

Anthony Gillioz, Jacky Casas, Elena Mugellini, and Omar Abou Khaled. 2020. Overview of the transformer-based models for nlp tasks. In *2020 15th Conference on computer science and information systems (FedCSIS)*, pages 179–183. IEEE.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. 2023. Diffuseq-v2: Bridging discrete and continuous text spaces for accelerated seq2seq diffusion models. *arXiv preprint arXiv:2310.05793*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. *Advances in neural information processing systems*, 32.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Shankar Kumar and Bill Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 169–176.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*.

Jason Lee, Raphael Shu, and Kyunghyun Cho. 2020. Iterative refinement in the continuous space for non-autoregressive neural machine translation. *arXiv preprint arXiv:2009.07177*.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343.

Yifan Li, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Diffusion models for non-autoregressive text generation: A survey. *arXiv preprint arXiv:2303.06574*.

Shan Liu, Yafang Zheng, Lei Lin, Yidong Chen, and Xiaodong Shi. 2023. A novel pos-guided data augmentation method for sign language gloss translation. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 392–403. Springer.

Sarfaraz Masood, Adhyan Srivastava, Harish Chandra Thuwal, and Musheer Ahmad. 2018. Real-time sign language gesture (word) recognition from video sequences using cnn and rnn. In *Intelligent Engineering Informatics: Proceedings of the 6th International Conference on FICTA*, pages 623–632. Springer.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. 2018. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*.

Amit Moryossef, Kayo Yin, Graham Neubig, and Yoav Goldberg. 2021. Data augmentation for sign language gloss translation. *arXiv preprint arXiv:2105.07476*.

Achraf Othman and Mohamed Jemni. 2012. English-asl gloss parallel corpus 2012: Aslg-pc12. In *5th Workshop on the Representation and Processing of Sign Languages: Interactions between Corpus and Lexicon LREC*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2021. Guiding non-autoregressive neural machine translation decoding with reordering information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13727–13735.

Razieh Rastgoo, Kourosh Kiani, and Sergio Escalera. 2021. Sign language recognition: A deep survey. *Expert Systems with Applications*, 164:113794.

Richard Sinkhorn and Paul Knopp. 1967. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985. PMLR.

MF Tolba and AS Elons. 2013. Recent developments in sign language recognition systems. In *2013 8th International Conference on Computer Engineering & Systems (ICCES)*, pages xxxvi–xlii. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in neural information processing systems*, 28.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Kayo Yin and Jesse Read. 2020. Better sign language translation with stmc-transformer. *arXiv preprint arXiv:2004.00588*.

A Sinkhorn Layer Formulation

The Sinkhorn layer transforms an input matrix $M \in \mathbb{R}^{n \times m}$ into an approximately doubly stochastic matrix P, where each row and column sums to 1. The transformation is defined as:

$$P = \text{Sinkhorn}(M, \tau, T)$$

where:

- M is the input score or cost matrix,
- τ is the temperature parameter controlling sharpness,
- T is the number of Sinkhorn iterations.

Step 1: Initialization (Softmax)

First, apply a softmax-like transformation to ensure non-negativity:

$$K = \exp\left(\frac{M}{\tau}\right)$$

Step 2: Iterative Normalization

For $t = 1, 2, \ldots, T$, perform the following updates:

$$K \leftarrow \frac{K}{\sum_j K_{ij}} \quad \text{(Row normalization)}$$

$$K \leftarrow \frac{K}{\sum_i K_{ij}} \quad \text{(Column normalization)}$$

Step 3: Final Output

After T iterations, the output is:

$$P = K$$

which approximates a doubly stochastic matrix.

B DiffuSeq Objective Equations

There are objective functions for DiffuSeq.

$$\mathbf{z}_t = \mathbf{x}_t \oplus \mathbf{y}_t, \quad \text{for } t \in [0, T] \tag{9}$$

$$\mathcal{L}_{\text{VLB}} = \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{z}_0)} \Bigg[\underbrace{\log \frac{q(\mathbf{z}_T|\mathbf{z}_0)}{p_\theta(\mathbf{z}_T)}}_{\mathcal{L}_T} + \sum_{t=2}^{T} \underbrace{\log \frac{q(\mathbf{z}_{t-1}|\mathbf{z}_0, \mathbf{z}_t)}{p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)}}_{\mathcal{L}_{t-1}} + \underbrace{\log \frac{q_\phi(\mathbf{z}_0|\mathbf{w}^{x \oplus y})}{p_\theta(\mathbf{z}_0|\mathbf{z}_1)}}_{\mathcal{L}_0} - \underbrace{\log p_\theta(\mathbf{w}^{x \oplus y}|\mathbf{z}_0)}_{\mathcal{L}_{\text{round}}} \Bigg] \tag{10}$$

$$\begin{aligned} \min_\theta \mathcal{L}_{\text{simple}} &= \min_\theta \Bigg[\sum_{t=2}^{T} \|\mathbf{z}_0 - f_\theta(\mathbf{z}_t, t)\|^2 \\ &+ \|\text{EMB}(\mathbf{w}^{x \oplus y}) - f_\theta(\mathbf{z}_1, 1)\|^2 - \log p_\theta(\mathbf{w}^{x \oplus y}|\mathbf{z}_0) \Bigg] \\ &\rightarrow \min_\theta [\sum_{t=2}^{T} \|\mathbf{y}_0 - \tilde{f}_\theta(\mathbf{z}_t, t)\|^2 \\ &+ \|\text{EMB}(\mathbf{w}^y) - \tilde{f}_\theta(\mathbf{z}_1, 1)\|^2 \\ &+ \mathcal{R}(\|\mathbf{z}_0\|^2)] \end{aligned} \tag{11}$$

Estimations were conducted using the Machine-Learning Impact calculator presented in (Lacoste et al., 2019).

Models	# Parameters	$Batch_{train}$	$Batch_{test}$	# Epochs/# Steps	GPU_{train}	$GPU_{inference}$
$mBART_{cc25}$	610,851,840	8	16	7	1 L4	1 A100
$mT5_{large}$	1,229,581,312	16	16	9	1 A100	1 A100
KD_{small}	173,207,040	32	16	15	1 L4	1 A100
LevT	172,917,590	16	16	12	1 A100	1 A100
PointerLevT	556,959,062	16	16	10	1 A100	1 A100
Vanilla Diffusion	87,336,792	128	100	600,000	1 A100	1 A100
DiffuSeq	91,225,274	2048	100	50,000	1 A100	1 A100

Table 3: Models' setups

Case Study 1 - Authentic Sentence	
gloss	X-IT BE BEYOND DOUBT THAT PROPOSE LEGISLATION BE ATTACK ON DESC-HUMAN RIGHTS.
reference	it is beyond doubt that the proposed legislation is an attack on human rights.
mBART	it is beyond doubt that the proposed legislation is an attack on human rights.
mT5	it is beyond doubt that the proposed legislation is an attack on human rights.
KD-mBART	it ismena isyonyond theubt thathat theposes legislation beyonttack on humanirrippolicyman rightsights.
LevT	it be beyond doubt that propose legislation be attack on desc-human rights.
PointerLevT	it be beyond doubt that propose legislation be attack on human rights.
DiffuSeq	it is beyond doubt that the proposed legislation is attacks on human rights.
Case Study 2 - Artificial Sentence	
gloss	Thompson Taylor professor political former U.S. science State Oklahoma Representative Carolyn state
reference	Carolyn Thompson Taylor is a former State Representative and professor of political science from the U.S. state of Oklahoma.
mBART	Carolyn Thompson is a former State professor of political science at Oklahoma U.S. Representative State University.
mT5	Carolyn Taylor Thompson is a former professor of political science at the U.S.
KD-mBART	Carol was is of science of.S. statesstiveyn'.
LevT	Thompson Taylor professor political former U.S. science State Oklahoma Representative Carolyn state.
PointerLevT	Carolyn Thompson Taylor former Representative professor U.S. political science State Oklahoma.
Diffusion-LM	of the loss in a and a further - wide development of the industry.
DiffuSeq	thompson is political representative was a science of representative of the professor of u. former. carolyn s the state of state.

Table 4: Predicted Outputs Generated by Models based on one authentic test data and one artificial test data