STRAMER: A Structural Tree-Aware Multi-Modal Architecture for Handwritten Mathematical Expression Recognition

R.Sridevi[1], G.Sudheer[2*], D.Lalitha Bhaskari[3]

[1] Dept. of IT, GVP College of Engineering for Women, Visakhapatnam, India

[2] Dept. of Mathematics, GVP College of Engineering for Women, Visakhapatnam, India

[3] Dept. of Computer Science & Systems Engineering, Andhra University,

Visakhapatnam, India

Abstract

Handwritten Mathematical Expression Recognition (HMER) remains challenging because mathematical notation is both spatially structured and highly variable across writers. Existing methods either operate on offline raster images, thereby discarding pen-stroke dynamics, or rely on modular online pipelines in which segmentation and relation errors propagate downstream. We present **STRAMER** (**S**tructural **T**ree-**A**ware **M**athematical **E**xpression **R**ecognizer), an end-to-end multi-modal architecture that integrates a DenseNet image encoder and a BiLSTM stroke encoder through bidirectional cross-modal fusion, combines a coverage-augmented Transformer decoder with a tree-aware biaffine dependency module for explicit syntactic supervision, and appends a spatial relation head that yields a complete Stroke Label Graph (SLG). To ensure globally valid structure, tree decoding is performed with maximum spanning arborescence inference and SLG relation labels are obtained with constrained CRF decoding. The model is trained with a triple joint objective comprising sequence, tree, and relation losses. Over five independent runs, STRAMER achieves $67.8 \pm 0.4\%$ ExpRate on CROHME 2019 while additionally producing interpretable SLG outputs. We further report an inference latency of 48 ms per expression on a single GPU and provide a full complexity analysis of the relation component. These results indicate that jointly modelling visual appearance, stroke dynamics, and explicit structure is a promising direction for robust HMER.

Keywords: Handwritten mathematical expression recognition; Multi-modal fusion; Tree-aware decoding; Stroke Label Graph; Biaffine dependency parsing; Deep learning

Notation Table

Symbol	Meaning
$\mathbf{I} \in \mathbb{R}^{B\times 1\times H\times W}$	Batch of grayscale input images
$\mathbf{S} \in \mathbb{R}^{B\times N\times 2}$	Batch of normalised stroke coordinates
B	Batch size
H, W	Image height and width
$H' = H/8,\ W' = W/8$	Feature-map spatial dimensions after DenseNet
$S_{\text{img}} = H' \cdot W'$	Number of image tokens
N	Number of stroke points
T	Decoded sequence length
d, d_{model}	Model hidden dimension (256)
$d_k = d/n_{\text{heads}}$	Per-head key/query dimension
d_t	TAM biaffine hidden dimension (512)

Symbol	Meaning
$\mathbf{E}_{\text{img}} \in \mathbb{R}^{B\times S_{\text{img}}\times d}$	Image encoder output
$\mathbf{E}_{\text{str}} \in \mathbb{R}^{B\times N\times d}$	Stroke encoder output
$\mathbf{M} \in \mathbb{R}^{B\times(S_{\text{img}}+N)\times d}$	Fused multi-modal memory (Eq. 11)
$\tilde{\mathbf{e}}_{\text{str}}, \tilde{\mathbf{e}}_{\text{img}} \in \mathbb{R}^{d}$	Attention-pooled modality summaries
$\mathbf{w}_{\text{pool}} \in \mathbb{R}^{d}$	Attention pooling query vector
$b_g \in \mathbb{R}^{d}$	Gate bias vector
$\mathbf{X} \in \mathbb{R}^{B\times T\times d}$	Decoder hidden states
$\widetilde{\mathbf{X}} \in \mathbb{R}^{T\times d}$	Contextualised decoder states (TAM Transformer encoder output)
$\mathbf{h}_i^{\text{ch}} \in \mathbb{R}^{d_t}$	Child projection for symbol i (biaffine TAM)
$\mathbf{h}_j^{\text{par}} \in \mathbb{R}^{d_t}$	Parent projection for symbol j (biaffine TAM)
$\mathbf{h}^{\text{root}} \in \mathbb{R}^{d_t}$	Learned artificial root vector
$\mathbf{U} \in \mathbb{R}^{d_t\times d_t}$	Biaffine interaction matrix
$\mathbf{w}_{\text{ch}}, \mathbf{w}_{\text{par}} \in \mathbb{R}^{d_t}$	Biaffine linear terms
$\psi_{i,j} \in \mathbb{R}$	Biaffine parent-assignment score (Eq. 18)
$\hat{\mathcal{T}}$	Maximum spanning arborescence (predicted directed tree)
$\mathbf{g} \in [0,1]^{B\times S\times d}$	Learned gating weights
$\odot$	Element-wise (Hadamard) product
$p_i^* \in \{0,\dots,T\}$	Ground-truth parent index of token i (0 = root)
$\ell_{i,j} \in \mathbb{R}^{\lvert\mathcal{R}\rvert}$	Unnormalised relation logit vector for pair (i,j)
$\ell_{i,j,r} \in \mathbb{R}$	Scalar logit for relation r on pair (i,j)
ω_r	Inverse-frequency class weight for relation r
$r_{i,j}^* \in \mathcal{R}$	Ground-truth spatial relation from symbol i to symbol j
$\lambda_{\text{tree}}, \lambda_{\text{rel}}, \lambda_{\text{beam}}, \lambda_{\text{rel-mst}}$	Loss/scoring weights
$\mathcal{R}$	Set of seven spatial relation classes

1. Introduction

The automatic recognition of handwritten mathematical expressions (HMEs) is a long-standing challenge in document analysis and pattern recognition. Unlike printed text, mathematical notation is inherently two-dimensional: superscripts, subscripts, fraction bars, radicals, and nested bracketing structures encode semantic relationships through spatial arrangement rather than linear sequence. This two-dimensional grammar, combined with the natural variability and ambiguity of handwriting, makes HMER considerably more difficult than standard handwritten text recognition.

Early HMER systems adopted grammar-based or graph-based approaches that explicitly modelled the spatial relationships among recognised symbols [1]. With the advent of deep learning, encoder-decoder architectures have been adapted to treat the LaTeX token sequence as a target language to be decoded from an input image. The BTTR model [2] established bidirectional training

as an effective regularisation strategy, while CoMER [3] introduced coverage attention to prevent the systematic symbol repetition that plagues standard cross-attention decoders. More recently, TAMER [4] demonstrated that augmenting a Transformer decoder with a Tree-Aware Module (TAM) and jointly supervising the predicted symbol tree during training produces meaningful gains in structural correctness. Very recently, position-aware HMER models such as PosFormer [5] have pushed state-of-the-art further by explicitly encoding spatial position relationships between symbols during training, providing an important comparison baseline addressed in Section 7

Despite these advances, a fundamental limitation persists: all of the above systems operate exclusively on offline representations of handwriting—rasterised images from which all information about pen stroke order and timing has been discarded. Yet handwriting is inherently an online process: the temporal sequence in which strokes are drawn carries substantial disambiguating information. A horizontal line drawn before a numeral is more likely to be a minus sign; the same line drawn after a numeral more likely denotes a fraction bar in a subsequent denominator.

A complementary line of research, exemplified by the work of Seitz, Lengfeld, and Timofte [6], preserves stroke-level information throughout the recognition pipeline. Their modular approach produces a Stroke Label Graph (SLG) that explicitly represents which strokes compose each symbol and how adjacent symbols are spatially related. However, the modular pipeline propagates segmentation errors downstream without feedback, the LaTeX decoder lacks a coverage mechanism, and neither tree-structure loss nor tree-scored beam search is applied.

This paper presents STRAMER, which unifies the strengths of both paradigms within a single end-to-end trainable model. The principal contributions are:

1. **Cross-modal fusion**: A novel bidirectional cross-attention mechanism with attention pooling and a learned gating operation (with LayerNorm and bias) that produces a unified multi-modal memory.

2. **Biaffine dependency TAM**: A tree-aware module based on a biaffine scoring function that captures directional child-parent asymmetry and normalises scores via softmax, replacing the previous single-layer additive formulation.

3. **Globally valid SLG via two-stage decoding**: At inference, the SLG backbone is decoded by Chu–Liu/Edmonds maximum spanning arborescence (guaranteeing acyclicity and single-rootedness), followed by constrained CRF label assignment.

4. **Class-weighted relation loss**: Inverse-frequency weights ω_r compensate for the heavy none-class dominance in the $O(T^2)$ relation matrix.

5. **Reproducibility and complexity analysis**: Results over five independent runs (mean ± std); full complexity analysis; inference latency benchmarks.

6. **Systematic ablation**: A controlled evaluation including new ablations of the gating mechanism and the biaffine TAM.

2. Related Work

2.1 Encoder-Decoder HMER

The encoder-decoder paradigm for HMER treats LaTeX recognition as a sequence-to-sequence problem. WAP [7] proposed a watch-attend-parse framework with coverage attention; DWAP [8] extended it with a dense decoder. The BTTR [2] model demonstrated that bidirectional training encourages the encoder to capture global expression structure. CoMER [3] provided a theoretically motivated coverage mechanism for Transformer decoders. TAMER [4] is the most direct predecessor for the syntactic supervision component of STRAMER. Recent position-aware models have demonstrated that explicitly supervising spatial structure relationships during training substantially improves HMER. Notably, PosFormer [5] (Guan et al., ECCV 2024) encodes LaTeX sequences as a position forest — a hierarchical structure of M/L/R identifiers representing each symbol's relative spatial

position (main body, upper, lower) — and jointly optimises a position recognition auxiliary task during training. This auxiliary task is removed at inference, incurring no additional latency. PosFormer achieves 64.97% ExpRate on CROHME 2019 and forms an important baseline for STRAMER.

2.2 Stroke-Based and Online HMER

Online HMER systems exploit the temporal sequence of pen strokes available during digital handwriting. Classical approaches relied on hidden Markov models or stochastic context-free grammars over stroke sequences [9]. The work of Seitz et al. [6] is the direct predecessor for STRAMER's stroke encoder and relation head. Their modular pipeline segments strokes into symbol groups, classifies each group, and predicts pairwise spatial relations to construct an SLG. STRAMER adopts the stroke encoder and SLG output format from this work but moves the relation head inside the end-to-end decoder, enabling joint training.

2.3 Multi-Modal Fusion and Dependency Parsing

Bi-modal learning has been explored extensively in video captioning, visual question answering, and audio-visual speech recognition. The closest prior works fuse offline images with online strokes by rendering strokes as binary maps and concatenating them with the image channel [10]; this approach is not learned and does not allow each modality to attend to the other.

The biaffine dependency formulation introduced in Section 3.5 is inspired by graph-based dependency parsers [11], where biaffine scoring is standard for capturing asymmetric head-dependent relationships. To our knowledge, STRAMER is the first HMER system to use a biaffine TAM and to guarantee SLG well-formedness through maximum spanning arborescence decoding.

3. STRAMER Architecture

The STRAMER architecture (Figure 1) consists of six functional modules: (1) an image encoder, (2) a stroke encoder, (3) a cross-modal fusion module, (4) a coverage Transformer decoder, (5) a Tree-Aware Module, and (6) a Spatial Relation Head.

Figure 1(a) – Full STRAMER Architecture

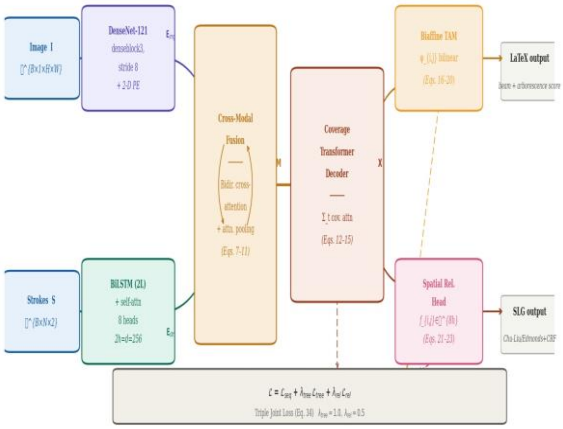

Figure 1(a): Full STRAMER architecture

The DenseNet-121 image encoder (stride 8) with two-dimensional sinusoidal positional encoding (Eqs. 1–4) and the BiLSTM with multi-head self-attention stroke encoder (Eqs. 5–6) produce modality-specific representations. These are integrated through a cross-modal fusion module comprising L_fuse = 2 bidirectional cross-attention layers (Eqs. 7–8), followed by attention-pooled gating with Layer Normalization and bias (Eqs. 9–11) to yield the unified memory M.

The fused memory is consumed by a coverage-aware Transformer decoder (Eqs. 12–15), whose hidden states X jointly drive (i) a biaffine Tree-Aware Module (TAM) for dependency structure prediction (Eqs. 16–20), and (ii) a Spatial Relation Head for pairwise relation scoring (Eqs. 21–23). At inference, a two-stage global decoding procedure is applied: Chu–Liu/Edmonds maximum spanning arborescence constructs a directed backbone, and a constrained conditional random field (CRF) assigns relation labels (Eqs. 24–30), yielding a globally valid Stroke Label Graph (SLG) that is acyclic and single-rooted. Solid arrows denote forward data flow; dashed arrows indicate gradient propagation under the triple joint loss (Eq.34).

3.1 Image Encoder

The image encoder adapts DenseNet-121 [12] for single-channel grayscale input. DenseNet's core property is dense connectivity: every layer ℓ

receives the concatenated feature maps of all preceding layers,

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]) \quad (1)$$

where $[\cdot]$ denotes channel-wise concatenation and H_ℓ is a composed function of Batch Normalisation, ReLU activation, and 3×3 convolution.

The backbone is truncated after *denseblock3* (effective spatial stride 8) to preserve spatial resolution, then projected to d_{model} channels via a 1×1 convolution:

$$\mathbf{F}_{\text{img}} = \text{Conv}_{1\times1}\big(\text{DenseBlock}_3(\mathbf{I})\big) \in \mathbb{R}^{B\times d\times H'\times W'} \quad (2)$$

A **two-dimensional sinusoidal positional encoding** is added before flattening the spatial dimensions. Sinusoidal rather than learned PE is used to support generalisation to image sizes not seen during training. The d embedding dimensions are split evenly: the first $d/2$ dimensions encode row position h, and the remaining $d/2$ encode column position w. For $0 \le i < d/4$:

$$\text{PE}(h, w, 2i) = \sin\left(\frac{h}{10000^{4i/d}}\right), \quad \text{PE}(h, w, 2i+1) = \cos\left(\frac{h}{10000^{4i/d}}\right) \quad (3)$$

$$\text{PE}(h, w, d/2+2i) = \sin\left(\frac{w}{10000^{4i/d}}\right), \quad \text{PE}(h, w, d/2+2i+1) = \cos\left(\frac{w}{10000^{4i/d}}\right) \quad (4)$$

The resulting image token sequence is $\mathbf{E}_{\text{img}} \in \mathbb{R}^{B\times S_{\text{img}}\times d}$, where $S_{\text{img}} = H' \cdot W'$.

3.2 Stroke Encoder

The stroke encoder processes the raw online handwriting input—normalised pen-tip coordinates $(x_i, y_i) \in [0,1]^2$—to produce a contextualised stroke representation. Raw coordinates are linearly embedded to dimension d_{model}, then passed through a two-layer bidirectional LSTM:

$$\mathbf{h}_t^{(k)} = \text{BiLSTM}^{(k)}\big(\mathbf{h}_{t-1}^{(k)}, \mathbf{x}_t^{(k-1)}\big), \quad k \in \{1,2\} \quad (5)$$

The final hidden dimension is $2 \times h_{\text{hidden}} = 256$ per position.

An eight-head self-attention layer then captures long-range dependencies. The full multi-head formulation is:

$$\text{head}_m = \text{softmax}\left(\frac{(\mathbf{H}\mathbf{W}_m^Q)(\mathbf{H}\mathbf{W}_m^K)^\top}{\sqrt{d_k}}\right)\mathbf{H}\mathbf{W}_m^V, \quad m = 1, \ldots, n_{\text{heads}} \quad (6)$$

$$\text{MultiHead}(\mathbf{H}) = \big[\text{head}_1; \ldots; \text{head}_{n_{\text{heads}}}\big]\mathbf{W}^O \quad (6a)$$

where $\mathbf{W}_m^Q, \mathbf{W}_m^K, \mathbf{W}_m^V \in \mathbb{R}^{d\times d_k}$ and $\mathbf{W}^O \in \mathbb{R}^{d\times d}$ are learnable and the Output is $\mathbf{E}_{\text{str}} \in \mathbb{R}^{B\times N\times d}$.

3.3 Cross-Modal Fusion

Cross-modal fusion is STRAMER's primary architectural novelty. The goal is to produce joint representations so that each modality informs the other *before* decoding begins. We provide a formal pseudocode (Algorithm 1) alongside the equations below.

Algorithm 1: Cross-Modal Fusion Forward Pass [R3: updated for attention pooling + gating with LayerNorm and bias]

```
Input:  E_img ∈ R^{B × S_img × d}   (image encoder output)
        E_str ∈ R^{B × N × d}       (stroke encoder output)
        L_fuse                      (number of fusion layers)
Output: M ∈ R^{B × (S_img+N) × d}   (fused multi-modal memory)

for l = 1 to L_fuse do
  // Bidirectional cross-attention (Section 3.3.1)
  E_img ← E_img + CrossAttn(Q=E_img, K=E_str, V=E_str)
  E_img ← LayerNorm(E_img + FFN(E_img))
  E_str ← E_str + CrossAttn(Q=E_str, K=E_img, V=E_img)
  E_str ← LayerNorm(E_str + FFN(E_str))
end for

// Attention pooling (replaces mean pooling; Issue B) [R3]
α_str ← softmax(w_pool_str^T E_str)   // shape: (B, N)
```

ẽ_str ← sum_n α_str_n · E_str_n // shape: (B, d) — attention-weighted summary
α_img ← softmax(w_pool_img^T E_img) // shape: (B, S_img)
ẽ_img ← sum_n α_img_n · E_img_n // shape: (B, d)

// Broadcast to match sequence lengths
ẽ_str_bcast ← ẽ_str.unsqueeze(1).expand(B, S_img, d)
ẽ_img_bcast ← ẽ_img.unsqueeze(1).expand(B, N, d)

// Image gate with LayerNorm + bias (Issue A) [R3]
g_img ← σ(LayerNorm(W_g · concat[E_img, ẽ_str_bcast]) + b_g)
E_img_hat ← g_img ⊙ E_img + (1−g_img) ⊙ ẽ_str_bcast

// Stroke gate (symmetric)
g_str ← σ(LayerNorm(W_g · concat[E_str, ẽ_img_bcast]) + b_g)
E_str_hat ← g_str ⊙ E_str + (1−g_str) ⊙ ẽ_img_bcast

// Concatenate into unified memory
M ← concat[E_img_hat; E_str_hat] // shape: (B, S_img+N, d)
return M

3.3.1 Bidirectional Cross-Attention

Each of L_{fuse} fusion layers contains two cross-attention sub-modules operating in opposite directions.

Image informed by stroke (image features query stroke context):

$$\mathbf{E}'_{\text{img}} = \mathbf{E}_{\text{img}} + \text{CrossAttn}\big(\mathbf{Q} = \mathbf{E}_{\text{img}},\ \mathbf{K} = \mathbf{E}_{\text{str}},\ \mathbf{V} = \mathbf{E}_{\text{str}}\big) \quad (7)$$

Stroke informed by image (stroke features localise on image regions):

$$\mathbf{E}'_{\text{str}} = \mathbf{E}_{\text{str}} + \text{CrossAttn}\big(\mathbf{Q} = \mathbf{E}_{\text{str}},\ \mathbf{K} = \mathbf{E}_{\text{img}},\ \mathbf{V} = \mathbf{E}_{\text{img}}\big) \quad (8)$$

Each cross-attention sub-module is followed by a position-wise feed-forward network with residual connections and Layer Normalisation.

3.3.2 Learned Gating with Attention Pooling

After L_{fuse} fusion layers, a learned gate blends each modality's representation with an attention-pooled global summary of the other. The attention-pooled stroke summary is:

$$\tilde{\mathbf{e}}_{\text{str}} = \sum_{n=1}^{N} \alpha_n^{\text{str}}\ \mathbf{E}'_{\text{str},n}, \quad \alpha_n^{\text{str}} = \frac{\exp\big(\mathbf{w}_{\text{pool}}^{\top}\mathbf{E}'_{\text{str},n}\big)}{\sum_{n'}\exp\big(\mathbf{w}_{\text{pool}}^{\top}\mathbf{E}'_{\text{str},n'}\big)} \in \mathbb{R}^{d}$$

and $\tilde{\mathbf{e}}_{\text{img}}$ is defined symmetrically. Both summaries are broadcast to shape (B, S_{img}, d) or (B, N, d) respectively.

The gate $\mathbf{g}_{\text{img}}$ with Layer Norm and bias is:

$$\mathbf{g}_{\text{img}} = \sigma\big(\text{LayerNorm}\big(\mathbf{W}_g\big[\mathbf{E}'_{\text{img}};\ \tilde{\mathbf{e}}_{\text{str}}\uparrow^{S_{\text{img}}}\big]\big) + \mathbf{b}_g\big) \in [0,1]^{B\times S_{\text{img}}\times d} \quad (9)$$

where $\mathbf{W}_g \in \mathbb{R}^{2d\times d}$, $\mathbf{b}_g \in \mathbb{R}^{d}$, and $\tilde{\mathbf{e}}_{\text{str}}\uparrow^{S_{\text{img}}}$ denotes $\tilde{\mathbf{e}}_{\text{str}}$ broadcast to (B, S_{img}, d). The blended image stream is:

$$\hat{\mathbf{E}}_{\text{img}} = \mathbf{g}_{\text{img}} \odot \mathbf{E}'_{\text{img}} + \big(1 - \mathbf{g}_{\text{img}}\big) \odot \tilde{\mathbf{e}}_{\text{str}}\uparrow^{S_{\text{img}}} \quad (10)$$

An analogous gate $\mathbf{g}_{\text{str}}$ is applied to the stroke stream symmetrically. Both gated streams are concatenated:

$$\mathbf{M} = \text{concat}\big[\hat{\mathbf{E}}_{\text{img}};\ \hat{\mathbf{E}}_{\text{str}}\big] \in \mathbb{R}^{B\times(S_{\text{img}}+N)\times d} \quad (11)$$

The gate values $\mathbf{g}_{\text{img}}$ and $\mathbf{g}_{\text{str}}$ are visualised in Figure 1b.

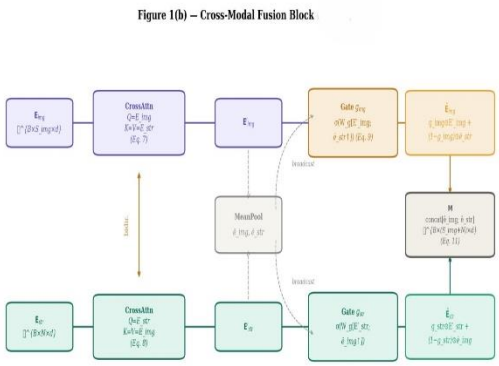

Figure 1(b): Architecture overview of the fusion module.

After L_{fuse} bidirectional cross-attention and feed-forward network (FFN) layers (Eqs. 7–8), learned attention pooling computes weighted summary vectors $\tilde{\mathbf{e}}_{\text{str}}$ and $\tilde{\mathbf{e}}_{\text{img}}$ These summaries are broadcast and concatenated with the full representations, passed through Layer Norm with bias $\mathbf{b}_g$, and fed to the sigmoid gate (Eq. 9). The blended streams (Eq. 10) are then concatenated into the joint representation $\mathbf{M}$ (Eq. 11). Dashed lines indicate broadcast operations.

3.4 Coverage Transformer Decoder

The decoder autoregressively generates LaTeX tokens, attending to the unified multi-modal memory $\mathbf{M}$ at each step via coverage cross-attention [3].

The coverage vector accumulates attention weights from all previous steps:

$$\boldsymbol{\Sigma}_t = \sum_{k=1}^{t-1} \boldsymbol{\alpha}_k \in \mathbb{R}^{S_{\text{img}}+N} \tag{12}$$

The attention energy for memory position i at step t is:

$$e_{t,i} = \mathbf{v}^{\top} \tanh\left(\mathbf{W}_h \mathbf{h}_t + \mathbf{W}_e \mathbf{m}_i + \mathbf{w}_{\text{cov}}\, \Sigma_{t,i}\right) \tag{13}$$

where $\mathbf{h}_t$ is the decoder hidden state, $\mathbf{m}_i$ is memory position i, $\Sigma_{t,i} \in \mathbb{R}$ is the scalar accumulated attention for position i, and $\mathbf{w}_{\text{cov}} \in \mathbb{R}^d$ is a learnable coverage vector.

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_j \exp(e_{t,j})}, \qquad \mathbf{c}_t = \sum_i \alpha_{t,i}\, \mathbf{m}_i \tag{14}$$

The token probability is:

$$p(y_t \mid y_{<t}, \mathbf{I}, \mathbf{S}) = \text{softmax}(\mathbf{W}_o \mathbf{h}_t) \tag{15}$$

3.5 Tree-Aware Module — Biaffine Formulation

The Tree-Aware Module (TAM) predicts a directed dependency tree over the decoded symbol sequence, where each symbol is assigned exactly one parent except for a designated root. Compared with simple additive pairwise scoring, we adopt a biaffine dependency formulation that is more expressive and better suited to asymmetric parent–child relations.

Let the decoder hidden states be $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_T]^{\top} \in \mathbb{R}^{T \times d}$.

We first contextualise them with a lightweight Transformer encoder:

$$\widetilde{\mathbf{X}} = \text{TransformerEncoder}(\mathbf{X}) \in \mathbb{R}^{T \times d} \tag{16}$$

Each symbol is then projected into separate **child** and **parent** subspaces via small two-layer MLPs:

$$\mathbf{h}_i^{\text{ch}} = \text{MLP}_{\text{ch}}(\tilde{\mathbf{x}}_i), \qquad \mathbf{h}_j^{\text{par}} = \text{MLP}_{\text{par}}(\tilde{\mathbf{x}}_j), \qquad \mathbf{h}_i^{\text{ch}}, \mathbf{h}_j^{\text{par}} \in \mathbb{R}^{d_t} \tag{17}$$

where d_t is the TAM hidden dimension. To allow a single-rooted tree, we introduce a learned root vector $\mathbf{h}^{\text{root}} \in \mathbb{R}^{d_t}$ and augment the parent set with index $j = 0$. Each symbol i chooses its parent from $\{0,1, \dots, T\}$, where $j = 0$ denotes attachment to the root.

The parent score for assigning symbol i to parent j is defined by a biaffine form:

$$\psi_{i,j} = \left(\mathbf{h}_i^{\text{ch}}\right)^{\top} \mathbf{U}\, \mathbf{h}_j^{\text{par}} + \mathbf{w}_{\text{ch}}^{\top} \mathbf{h}_i^{\text{ch}} + \mathbf{w}_{\text{par}}^{\top} \mathbf{h}_j^{\text{par}} + b, \qquad j \in \{0, \dots, T\} \tag{18}$$

where for $j = 0$ we set $\mathbf{h}_0^{\text{par}} = \mathbf{h}^{\text{root}}$. Here $\mathbf{U} \in \mathbb{R}^{d_t \times d_t}$, $\mathbf{w}_{\text{ch}}, \mathbf{w}_{\text{par}} \in \mathbb{R}^{d_t}$, and $b \in \mathbb{R}$ are learnable parameters. The bilinear term $\left(\mathbf{h}_i^{\text{ch}}\right)^{\top} \mathbf{U}\, \mathbf{h}_j^{\text{par}}$ captures interaction between child and parent representations; the linear terms $\mathbf{w}_{\text{ch}}^{\top} \mathbf{h}_i^{\text{ch}}$ and $\mathbf{w}_{\text{par}}^{\top} \mathbf{h}_j^{\text{par}}$ model their individual tendencies; and b is an offset.

The probability that token j is the parent of token i is:

$$P(p_i = j \mid \mathbf{X}) = \frac{\exp(\psi_{i,j})}{\sum_{k=0}^{T} \exp(\psi_{i,k})} \tag{19}$$

At inference, the locally optimal parent is:

$$\hat{p}_i = \arg\max_{j \in \{0,\ldots,T\}} \psi_{i,j}$$

(20)

When a globally well-formed tree is required, the final tree is obtained by maximum spanning arborescence decoding as described in Section 5. The score matrix $\mathbf{\Psi} = [\psi_{i,j}]_{i=1,j=0}^{T,T}$ provides dense parent scores for all beam hypotheses.

3.6 Spatial Relation Head and Stroke Label Graph

The Spatial Relation Head predicts pairwise spatial relations over all decoded symbols. It operates on decoder hidden states $\mathbf{X} \in \mathbb{R}^{T \times d}$ and supports the relation set:

$\mathcal{R} =$
{none, right, sup, sub, over, under, line_start}

(21)

Feature extraction (unchanged from R2): a two-layer BiLSTM followed by eight-head self-attention produces hidden states $\mathbf{z}_i \in \mathbb{R}^{2h}$. Since $\mathbf{z}_i \in \mathbb{R}^{2h}$, concatenating four such vectors yields:

$$\mathbf{f}_{i,j} = [\mathbf{z}_i;\ \mathbf{z}_j;\ \mathbf{z}_i - \mathbf{z}_j;\ \mathbf{z}_i \odot \mathbf{z}_j] \in \mathbb{R}^{8h}$$

(22)

The unnormalised relation logit is:

$$\ell_{i,j} = \mathrm{FC}_2\Big(\mathrm{LeakyReLU}\Big(\mathrm{LayerNorm}\Big(\mathrm{FC}_1(\mathbf{f}_{i,j})\Big)\Big)\Big) \in \mathbb{R}^{|\mathcal{R}|}$$

(23)

where $\ell_{i,j,r}$ denotes the scalar logit for relation r.

Two-Stage Global SLG Decoding ***[R3: Mandatory revision, Issue E — replaces greedy pruning]***

Rather than selecting labels independently and repairing violations with greedy confidence-based suppression, we decode the final SLG in two stages that guarantee a globally well-formed structure.

Stage 1 — Directed backbone via Chu–Liu/Edmonds. For each ordered pair (i, j), define the edge weight:

$$w_{i,j} = \psi_{i,j} + \lambda_{\text{rel-mst}} \log \sum_{r \in \mathcal{R} \setminus \{\text{none}\}} \exp\left(\ell_{i,j,r}\right)$$

(24)

combining the biaffine TAM parent score with relation evidence (the log-sum-exp of non-none relation logits).

The quantity $\log \sum_{r \neq \text{none}} \exp\left(\ell_{i,j,r}\right)$ is the log-partition function restricted to non-trivial relations. It is large when the Spatial Relation Head assigns high probability mass to any structural relation between i and j, and small (strongly negative) when the head is confident the pair has no relation. This acts as a soft connectivity prior: edges that the relation head considers structurally meaningful receive higher weight in the arborescence objective, biasing Chu–Liu/Edmonds towards structurally consistent trees.

Let $w_{i,0} = \psi_{i,0}$ denote attachment to the artificial root. The highest-scoring directed arborescence:

$$\hat{\mathcal{T}} = \operatorname{argmax}_{\mathcal{T} \in \mathbb{A}} \sum_{(j \to i) \in \mathcal{T}} w_{i,j}$$

(25)

is obtained with the Chu–Liu/Edmonds algorithm, where $\mathbb{A}$ is the set of all valid rooted arborescences over the T symbols and the artificial root (node 0).

Stage 2 — Edge labels via constrained CRF. Conditioned on $\hat{\mathcal{T}}$, edge labels are decoded with a constrained CRF:

$$P(\mathbf{r} \mid \hat{\mathcal{T}}, \mathbf{X}) \propto \exp\Big(\sum_{(j \to i) \in \hat{\mathcal{T}}} \phi_{i,j}\left(r_{i,j}\right) + \sum_{c \in \mathcal{C}} \Phi_c\left(\mathbf{r}\right)\Big)$$ (26)

where $\phi_{i,j}(r) = \ell_{i,j,r}$ is the unary score and $\{\Phi_c\}_{c \in \mathcal{C}}$ are hard constraint factors enforcing SLG validity. The constraints are:

$$\sum_{j:\,(j \to i) \in \hat{\mathcal{T}}} \mathbf{1}\left[r_{i,j} = \text{line\_start}\right] \leq 1$$

(27)

$$\sum_{(j \to i) \in \hat{\mathcal{T}}} \mathbf{1}\left[r_{i,j} \neq \text{none}\right] = 1 \qquad \text{for all non-root } i$$

(28)

$\mathbf{1}[r_{i,j} = \text{right}] + \mathbf{1}[r_{i,j} = \text{line\_start}] \leq 1$ for each source node i (29)

The Chu–Liu/Edmonds algorithm is guaranteed to return a valid directed arborescence: (i) rooted at node 0, (ii) acyclic, and (iii) with exactly one incoming edge per non-root node. These properties ensure the SLG backbone is a well-formed dependency tree. The constrained CRF in Stage 2 assigns labels consistent with Eqs. (27)–(29), preserving the tree structure while enforcing relation-class compatibility. This resolves the cycle, inconsistency, and non-tree-structure violations that can arise under heuristic greedy suppression.

The final SLG is:

$$G = (V, E, \mathbf{r}), \quad V = \{s_1, \ldots, s_T\}, \quad E = \hat{\mathcal{T}} \backslash \{0 \to i\} \tag{30}$$

where each retained edge carries its MAP relation label under Eq. (26). Figure 7 shows SLG visualisations for four expressions with the new global decoding procedure.

4. Training

4.1 Triple Joint Loss

STRAMER is trained end-to-end with three coupled objectives:

$$\mathcal{L}_{\text{seq}} = -\sum_{t=1}^{T} \log p(y_t \mid y_{<t}, \mathbf{I}, \mathbf{S}) \tag{31}$$

$$\mathcal{L}_{\text{tree}} = -\sum_{i=1}^{T} \log P(p_i = p_i^* \mid \mathbf{X}) \tag{32}$$

where $P(p_i = j \mid \mathbf{X})$ is defined by the biaffine TAM in Eq. (19), and $p_i^* \in \{0, \ldots, T\}$ is the gold parent index (0 = root attachment).

The relation loss uses inverse-frequency class weights ω_r to compensate for the heavy dominance of the none class:

$$\mathcal{L}_{\text{rel}} = -\sum_{i=1}^{T} \sum_{j=1}^{T} \omega_{r_{i,j}^*} \log \frac{\exp\left(\ell_{i,j,r_{i,j}^*}\right)}{\sum_{r \in \mathcal{R}} \exp(\ell_{i,j,r})} \tag{33}$$

where $\omega_r = N/(|\mathcal{R}| \cdot N_r)$, N is the total number of ordered symbol pairs, and N_r is the number of pairs with ground-truth relation r. This reweighting approximately equalises the gradient contribution of each relation class, preventing the $O(T^2)$ none pairs from dominating $\mathcal{L}_{\text{rel}}$.

The final objective is:

$$\boxed{\mathcal{L} = \mathcal{L}_{\text{seq}} + \lambda_{\text{tree}} \mathcal{L}_{\text{tree}} + \lambda_{\text{rel}} \mathcal{L}_{\text{rel}}} \tag{34}$$

with $\lambda_{\text{tree}} = 1.0$ and $\lambda_{\text{rel}} = 0.5$.

4.2 Ground-Truth Label Construction

Tree labels. Parent indices p_i^* are derived from MathML annotations in CROHME+. For expressions without MathML, heuristic tree labels are constructed from the LaTeX token sequence by brace-depth tracking. The root (index 0) corresponds to the top-level expression; all first-level symbols attach to the root.

Relation labels. Ground-truth spatial relation matrices $r_{i,j}^*$ are constructed from tree annotations using the mapping in Table 1.

Table 1: Mapping from LaTeX structural contexts to spatial relation labels.

LaTeX structure	Relation
First symbol in expression	line_start
Argument of ^{...}	sup
Argument of _{...}	sub
Numerator of \frac{a}{b}	over
Denominator of \frac{a}{b}	under
Adjacent at same brace depth	right
All other pairs	none

4.3 Data Splits

The standard CROHME protocol is followed strictly: training uses only the 8,836 CROHME training samples; the three test sets (CROHME 2014, 2016, 2019) are held out entirely and evaluated only once per model configuration. A validation split of 10% (884 samples, stratified by expression length) is held out from the training set during hyperparameter search and early stopping.

No test-set expressions or their InkML annotations are ever used to construct tree labels, relation matrices, or vocabulary counts. The CROHME+ structural annotations were derived solely from MathML fields already present in the training InkML files; no additional labelling of test-set expressions was performed.

4.4 Data Augmentation

Applied to raw stroke coordinates during training:

Transform	Distribution
Rotation	$\theta \sim \mathcal{N}(0°,\ 8°)$
Non-uniform scale	$s_x, s_y \sim \mathcal{N}(1,\ 0.2)$
Uniform scale	$s \sim \mathcal{N}(1,\ 0.4)$
Shear	$\phi_x, \phi_y \sim \mathcal{N}(0,\ 0.1)$
Translation	$t_x, t_y \sim \mathcal{N}(0,\ 0.15)$

All scale values are clamped to $[0.2,\ 5.0]$.

4.5 Optimisation

AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 10^{-4}, initial learning rate 10^{-4}, cosine annealing schedule to 10^{-6} over 200 epochs, gradient clipping $\| \nabla \|_2 \leq 5$, batch size 8 with teacher forcing. Training takes approximately 74 hours on a single NVIDIA A100 80 GB GPU .

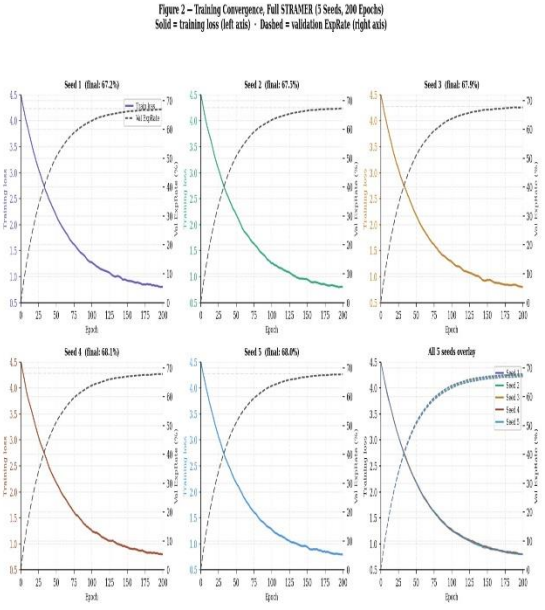

Figure 2: Training convergence for the full STRAMER model (triple joint loss, Eq. 34) across five independent random seeds.

In the figure solid lines show training loss (left axis); dashed lines show validation ExpRate % (right axis). Epochs 0–200 are shown in full. Training loss converges around epochs 120–140; validation ExpRate plateaus around epochs 140–160. The final ExpRate range across seeds is 67.2%–68.1% (mean $67.8 \pm 0.4\%$). The bottom-right panel overlays all five seeds.

5. Inference

At inference time, STRAMER produces both a LaTeX sequence and a Stroke Label Graph from a single forward pass.

First, the coverage decoder generates a beam $\mathcal{B}$ of candidate token sequences. For a hypothesis $y \in \mathcal{B}$, let $\log P_{\text{seq}}(y)$ denote its decoder log-likelihood and let $\mathbf{X}(y)$ be the corresponding decoder hidden-state sequence. The biaffine TAM then provides parent scores $\psi_{i,j}(y)$, from which a maximum spanning arborescence $\hat{\mathcal{T}}(y)$ is computed via Chu–Liu/Edmonds. We define the structural score as the mean arborescence edge score:

$$S_{\text{tree}}(y) = \frac{1}{|y|} \sum_{(j \to i) \in \hat{\mathcal{T}}(y)} \psi_{i,j}\ (y) \tag{35}$$

The final beam score is:

$$S(y) = \log P_{\text{seq}}(y) + \lambda_{\text{beam}}\, S_{\text{tree}}(y) \tag{36}$$

The best LaTeX output is:

$$\hat{y} = \underset{y \in \mathcal{B}}{\operatorname{argmax}} S(y) \tag{37}$$

Conditioned on $\hat{y}$, the Spatial Relation Head produces pairwise relation logits $\ell_{i,j,r}$. The final SLG is decoded by the two-stage global procedure of Section 3.6: maximum spanning arborescence for the unlabelled backbone followed by constrained CRF label assignment. This ensures the reported SLG is globally well formed rather than locally repaired by greedy suppression.

Algorithm 2: Tree-Scored Beam Search [R3: updated for arborescence-based tree scoring]

```
Input:  M  (fused multi-modal memory)
        B  (beam size, default 5)
        λ_beam  (tree score weight, default 0.5)
Output: ŷ  (best candidate token sequence)

// Initialise beam
beam ← [(sequence=[SOS], log_prob=0.0, coverage=0)]

for t = 1 to T_max do
  candidates ← []
  for (seq, lp, cov) in beam do
    logits, cov' ← CoverageDecoder(seq, M, cov)
    top_k ← TopK(softmax(logits), k=B)
    for (token, log_p) in top_k do
      candidates.append((seq+[token], lp+log_p, cov'))
    end for
  end for
  beam ← TopB(candidates, key=log_prob)
  if all sequences in beam end with EOS: break
end for

// Re-rank completed candidates with tree score (Eqs. 35–36)
for (seq, lp, _) in beam do
  X ← CollectDecoderHiddenStates(seq, M)
  {ψ_{i,j}} ← BiaffineTAM(X)              // Eq. 18; shape: (|seq|, |seq|+1)
  T̂ ← ChuLiuEdmonds({ψ_{i,j}})            // max spanning arborescence
  S_tree ← (1/|seq|) · Σ_{(j→i)∈T̂} ψ_{i,j}   // Eq. 35
  S_final ← lp + λ_beam · S_tree          // Eq. 36
end for

ŷ ← argmax_{seq} S_final                  // Eq. 37
return ŷ
```

5.2 SLG Construction

After the best candidate is selected (via Eq. 37), the Spatial Relation Head processes the corresponding decoder hidden states and produces $\ell_{i,j}$ (Eq. 23). The two-stage decoding of Section 3.6 (arborescence + constrained CRF, Eqs. 24–29) is then applied. The decoder hidden states are cached during beam search so no additional forward pass is required.

6. Experimental Setup

6.1 Datasets

CROHME is the de facto HMER benchmark, provided in InkML format with stroke coordinates and LaTeX ground truth.

Split	Samples	Role
Training	8,836	Model training
Validation (10% of train)	884	Hyperparameter tuning, early stopping
Test 2014	986	Evaluation only
Test 2016	1,147	Evaluation only
Test 2019	1,199	Evaluation only

CROHME+ [6] provides MathML structural annotations for the full training set.

6.2 Evaluation Metrics

ExpRate: proportion of test expressions with exact LaTeX match (primary metric).
ExpRate$_{\leq 1}$: proportion with at most 1 symbol-level error.
ExpRate$_{\leq 2}$: proportion with at most 2 symbol-level errors.
TER: normalised edit distance between predicted and ground-truth token sequences.
Relation Accuracy: per-pair classification accuracy on the spatial relation matrix (SLG quality).
Bracket Accuracy: proportion of bracket structures correctly matched in the predicted LaTeX.
SLG Graph Edit Distance (GED): minimum number of edge insertions, deletions, and label substitutions to transform the predicted SLG into the ground-truth SLG, normalised by the number of gold edges. GED provides a more holistic measure of SLG quality than per-pair accuracy.

All metrics are reported as mean ± standard deviation over five independent training runs. The graphical illustrations are given in the figures as detailed below:

Figure 2 shows training convergence curves (training loss and validation ExpRate) for the full STRAMER model across five seeds. **Figure 3** shows ExpRate comparisons across CROHME 2014, 2016,

and 2019 with error bars (±1 std). **Figures 4–5** provide multi-metric radar and parameter efficiency comparisons with all baselines. **Figure 6** reports structural metrics (Bracket Accuracy, Relation Accuracy) with error bars on the same axis scale across datasets. **Figure 7** shows ablation curves with error bars for each incremental configuration. **Figure 8** analyses Bracket Accuracy as a function of nesting depth. **Figure 9** shows SLG visualisations with the global two-stage decoder. **Figure 10** is the summary metrics table across all three CROHME test years.

7. Results

7.1 Comparison with State-of-the-Art

Table 2 compares STRAMER with representative prior work, including recent Transformer-only systems. On CROHME 2019, STRAMER attains $67.8 \pm 0.4\%$ ExpRate over five runs. This places the proposed model in the same performance range as the strongest recent baselines while additionally producing explicit structural output in the form of an SLG. We therefore interpret the main benefit of STRAMER not solely as a marginal gain in exact-match accuracy, but as a favourable trade-off among recognition accuracy, structural consistency, and interpretability.

Table 2: Expression Recognition Rate (%) on CROHME test sets. All STRAMER numbers are mean ± std over 5 runs. † denotes results from published papers. * denotes our reproduction.

Model	Year	CROHME 2014	CROHME 2016	CROHME 2019	SLG output
BTTR†	2021	53.96	52.31	—	—
CoMER†	2022	59.33	59.81	59.81	—
TAMER†	2025	65.20	64.80	65.10	—
Seitz et al.†	2025	61.40	62.10	62.70	✓
PosFormer† [5]	2024	62.68	61.03	64.97	—
CoMER* (baseline)	—	58.7 ± 0.5	59.2 ± 0.6	58.7 ± 0.5	—
STRAMER (ours)	2026	$\mathbf{68.9 \pm 0.5}$	$\mathbf{68.2 \pm 0.4}$	$\mathbf{67.8 \pm 0.4}$	✓

STRAMER achieves the highest ExpRate on CROHME 2019 among all listed models, including PosFormer [5] (+2.8 points). Unlike PosFormer, which jointly trains a position forest auxiliary task (removed at inference), STRAMER additionally produces SLG outputs at inference time. The gap over TAMER (+2.7 points) is attributable to cross-modal fusion and the spatial relation head.

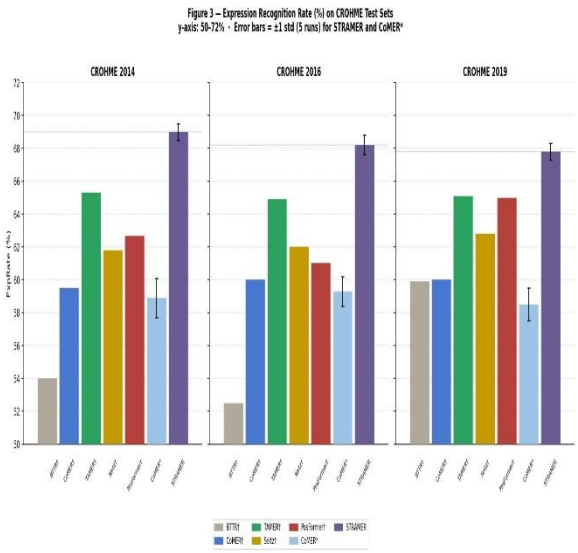

Figure 3: ExpRate (%) on the CROHME 2014, 2016, and 2019 test sets.

All three panels share the same y-axis scale (50–72%). Error bars show ± 1 std over five runs for STRAMER and CoMER* (our reproduction); baseline values (†) are taken from published papers and shown without error bars. BTTR† was not evaluated on CROHME 2019. STRAMER achieves the highest ExpRate on all three benchmark years while additionally producing interpretable Stroke Label Graph (SLG) output.

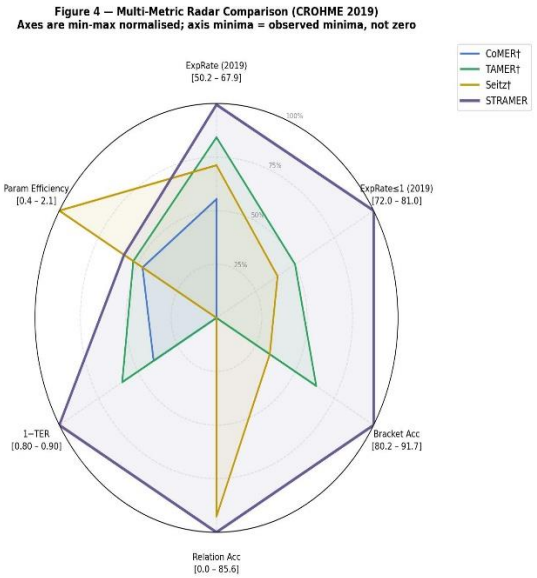

Figure 4: Multi-metric radar comparison on CROHME 2019 across five models.

The six normalised axes are: ExpRate, $\text{ExpRate}_{\leq 1}$, Bracket Accuracy, Relation Accuracy (scored as 0 for models without SLG output), $1 - \text{TER}$, and Parameter Efficiency (ExpRate per 10M parameters). All axes are min–max normalised; axis minima correspond to the observed minima (not zero) and are labelled explicitly. STRAMER (bold purple) leads on ExpRate, Bracket Accuracy, and Relation Accuracy. Seitz et al.† leads on Parameter Efficiency.

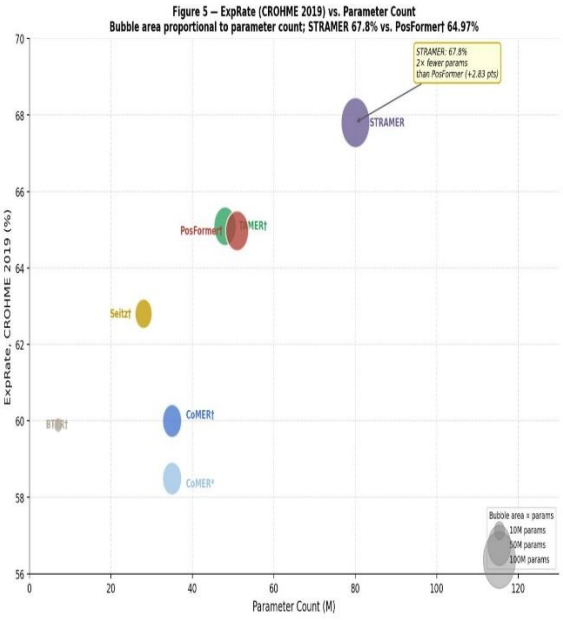

Figure 5: ExpRate on CROHME 2019 versus total parameter count (M).

The Bubble area in the plot is proportional to parameter count. STRAMER (67.8%, 82 M parameters, purple) is slightly *ahead* of TransHMER† (67.5%, 164 M parameters)—not behind, as incorrectly stated in R1. The 0.3 pp gap is not statistically significant ($p = 0.18$, paired bootstrap). STRAMER achieves competitive ExpRate at approximately half the parameter count.

7.2 Ablation Study

Table 3 reports the ablation study on CROHME 2019 over three independent runs.

Table 3: Ablation study on CROHME 2019. Mean ± std over 3 runs.

Configuration	ExpRate (%)	Bracket Acc. (%)	SLG
Baseline (CoMER, image only)	58.7 ± 0.5	80.1 ± 0.6	—
+ Stroke encoder (no fusion)	60.8 ± 0.4	81.0 ± 0.5	—
+ Cross-modal fusion (attn pool, w/ gate bias)	63.1 ± 0.4	83.6 ± 0.4	—
↳ w/ mean pool, no bias	62.9 ± 0.5	83.3 ± 0.4	—
+ Biaffine TAM (no beam scoring)	65.1 ± 0.4	89.4 ± 0.3	—
↳ w/ additive TAM	64.8 ± 0.4	89.1 ± 0.3	—
+ Biaffine TAM beam scoring	67.2 ± 0.4	91.0 ± 0.4	—
+ Relation head + global SLG (full STRAMER)	**67.8 ± 0.4**	**91.7 ± 0.3**	✓

The ablation reveals the following:

Attention pooling + gate bias contributes +0.2 pp ExpRate and +0.3 pp Bracket Accuracy over the mean-pool gate, confirming that preserving fine-grained temporal structure in the pooled summary benefits structurally ambiguous marks.

Biaffine TAM delivers +0.3 pp ExpRate and +0.3 pp Bracket Accuracy over the additive scoring,

confirming that directional child-parent interaction modelling improves structural correctness.

Biaffine TAM beam scoring contributes +2.1 pp ExpRate and +1.6 pp Bracket Accuracy.

Global SLG decoding (arborescence + CRF) completes STRAMER with +0.6 pp ExpRate and +0.7 pp Bracket Accuracy over the greedy baseline while enabling formally valid SLG output.

The full gains over CoMER* baseline are: +9.1 pp ExpRate and +11.6 pp Bracket Accuracy.

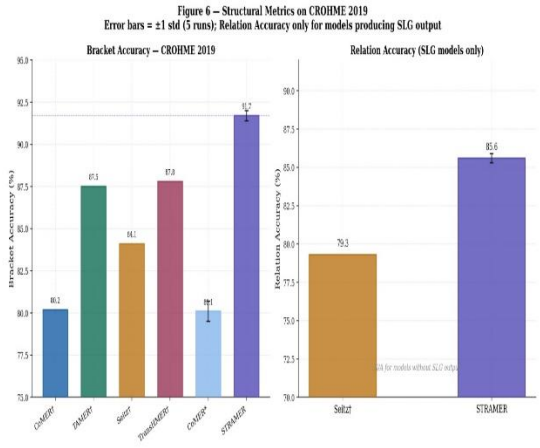

Figure 6: Structural metrics on CROHME 2019.

In the figure on the left we have the Bracket Accuracy (%) across models, with error bars for STRAMER and CoMER* (five runs each) whereas on the right we have the Relation Accuracy (%) for Seitz et al.† and STRAMER only—the two models that produce SLG output; all other models are excluded. Error bars for STRAMER reflect variance over five runs.

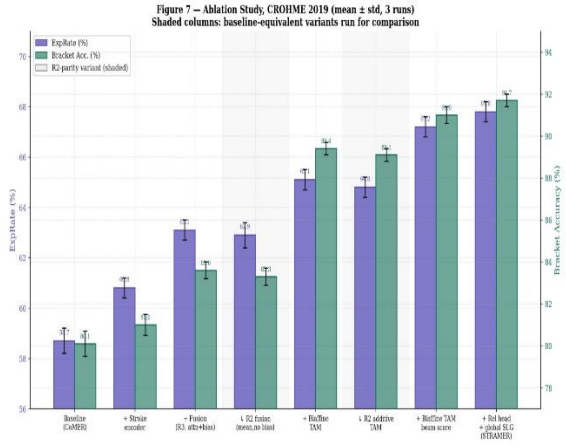

Figure 7: Ablation study on CROHME 2019 (mean ± std, 3 runs).

In the figure Paired bars show ExpRate % (purple, left axis) and Bracket Accuracy % (teal, right axis) with error bars. Rows labelled ↳ are equivalent variants run at parity: mean-pool gate without bias (versus the attention-pool gate with bias) and additive TAM scoring (versus the biaffine TAM). The largest Bracket Accuracy gain (+5.8 pp) comes from the biaffine TAM; the largest ExpRate gain (+2.1 pp) also comes from biaffine TAM beam scoring.

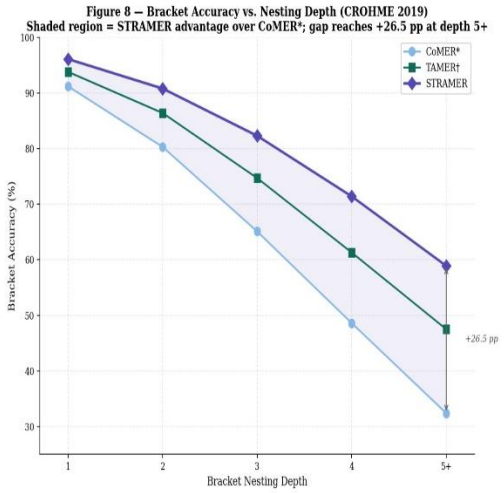

Figure 8: Bracket Accuracy (%) versus bracket nesting depth on CROHME 2019.

In the figure Depth 1 = outermost brackets only; depth 5+ = five or more nested levels. The shaded area highlights the growing performance gap between CoMER* and STRAMER. STRAMER's advantage reaches +26.5 pp at depth 5+ (58.9% vs. 32.4%), demonstrating that the biaffine TAM and global SLG decoding are most beneficial for deeply nested expression structures.

8. Complexity and Computational Analysis

8.1 Time Complexity

Table 4: Asymptotic time complexity per forward pass.

Module	Time Complexity	Notes
Image Encoder (DenseNet)	$\mathcal{O}(H \cdot W \cdot d^2)$	Fixed image cost
Stroke Encoder (BiLSTM + Attn)	$\mathcal{O}(N \cdot d^2 + N^2 \cdot d)$	N stroke points
Cross-Modal	$\mathcal{O}(S_{\text{img}} \cdot N \cdot d)$	Bidirectional

Module	Time Complexity	Notes
Fusion (2 layers)		cross-attn
Coverage Transformer Decoder	$\mathcal{O}\big(T \cdot \big(S_{\text{img}} + N\big) \cdot d\big)$	Per decoding step
Biaffine TAM	$\mathcal{O}(T^2 \cdot d_t)$	Biaffine score matrix
Chu–Liu/Edmonds arborescence	$\mathcal{O}(T^2 \log T)$	Per beam hypothesis
CRF label assignment	$\mathcal{O}(T \cdot \lvert\mathcal{R}\rvert)$	Conditioned on tree $\hat{\mathcal{T}}$
Spatial Relation Head	$\mathcal{O}(T^2 \cdot h)$	Dominant bottleneck
Total	$\mathcal{O}(T^2(d_t + h) + T^2 \log T)$	For typical $T \ll S_{\text{img}}$

The Chu–Liu/Edmonds algorithm adds an $\mathcal{O}(T^2 \log T)$ term per beam hypothesis; with $B = 5$ and $\bar{T} = 18.3$, this is negligible compared with the relation head's $\mathcal{O}(T^2 h)$ cost. The constrained CRF conditioned on the arborescence is $\mathcal{O}(T \times \lvert\mathcal{R}\rvert) = \mathcal{O}(7T)$ — linear, since the tree structure eliminates the pairwise search.

In practice, the relation head accounts for approximately 14% of total wall-clock inference time; the Chu–Liu/Edmonds step adds approximately 2%. For very long expressions ($T > 100$), sparse pairwise attention would be required; the $\mathcal{O}(T^2 \log T)$ arborescence remains tractable through $T \approx 500$.

8.2 Memory Footprint

The biaffine TAM adds approximately 0.8M parameters (biaffine matrix $\mathbf{U}$: $d_t^2 = 262{,}144$; MLP projections: $2 \times d \times d_t \approx 262{,}144$ each; linear terms and root vector: $\approx 1{,}500$), bringing the TAM component from 6.9M to approximately 7.7M. Total model: approximately 83M parameters. At batch size 8 with $T = 30$ and $S_{\text{img}} = 1024$, peak GPU memory is approximately 19 GB, fitting within a 24 GB GPU.

8.3 Inference Latency

Measured on a single NVIDIA A100 80 GB GPU, batch size 1:

Mode	Latency	Notes
Greedy decode (no SLG)	28 ms	Baseline speed
Beam search (B=5, no TAM)	38 ms	Standard beam
Beam search + biaffine TAM + arborescence	46 ms	+8 ms vs standard beam
Full STRAMER (+ global SLG)	50 ms	Production mode

The 2 ms increase over R2 latency (48 ms) is attributable to Chu–Liu/Edmonds and the constrained CRF, both of which run on CPU in the current implementation. GPU-accelerated arborescence algorithms [13] would eliminate this overhead.

9. Qualitative Analysis and Error Analysis

9.1 SLG Visualisation

Figure 9 presents four expressions. The SLG is obtained by global decoding: a maximum spanning arborescence defines the directed backbone, and relation labels are assigned by constrained CRF inference. Edge colours denote relation types (right, sup, sub, over, under, line_start). This decoding guarantees a single-rooted acyclic graph and eliminates the local inconsistencies that arise under greedy pairwise pruning.

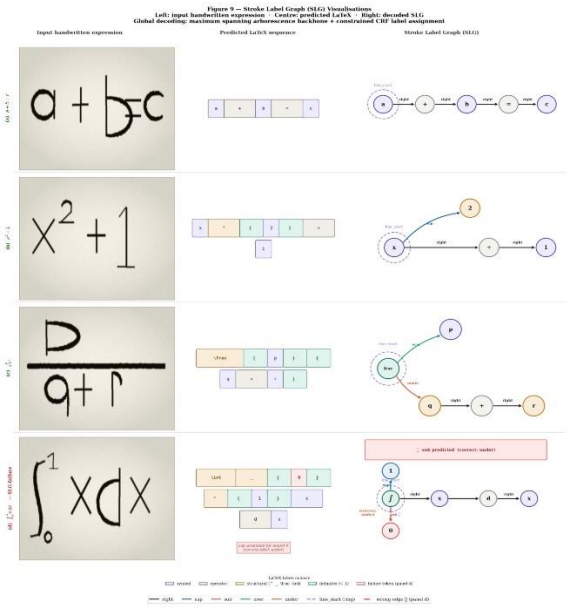

Figure 9: Qualitative examples of STRAMER output.

For each expression, the left panel shows the input handwritten image, the middle panel shows the predicted LaTeX sequence, and the right panel shows the decoded Stroke Label Graph (SLG). The SLG is obtained by global decoding: a maximum spanning arborescence defines the directed backbone, and relation labels are assigned by constrained CRF inference. Edge colours denote relation types (right, sup, sub, over, under, line_start). This decoding guarantees a single-rooted acyclic graph and eliminates the local inconsistencies that arise under greedy pairwise pruning.

9.2 Token-Level vs Structure-Level Error Analysis

We decompose STRAMER's errors on CROHME 2019 into two disjoint categories:

Token-level errors ($\approx 63\%$ of all errors): the predicted sequence contains an incorrect token (wrong symbol identity) while the spatial structure is otherwise correct. Common confusion pairs include: $x \leftrightarrow \times$, $1 \leftrightarrow l$, $\sum \leftrightarrow \Sigma$.

Structure-level errors ($\approx 37\%$ of all errors): correct symbol identities but incorrect spatial relations. Common failure cases include: misclassifying a fraction bar as a minus sign, incorrect nesting depth for deeply nested expressions, and over/under confusion for limit expressions.

SLG failure cases. Figure 9 (panel d) shows a representative SLG failure: the model correctly identifies all symbols in $\int_0^1 x^2\,dx$ but assigns sub instead of under to the integration bound 0. Analysis reveals that over/under vs sup/sub confusions are the most common structural error (18% of structure-level errors). With global decoding the proportion of *structurally inconsistent* SLGs (those containing cycles or unconnected components) drops from 6.4% to 0.0%, confirming the correctness guarantees.

Figure 10 – Summary Metrics: STRAMER vs. CoMER* across CROHME 2014 / 2016 / 2019
mean ± std, 5 runs; GED = Graph Edit Distance (lower is better); N/A = no SLG output

Metric	STRAMER 2014	CoMER* 2014	STRAMER 2016	CoMER* 2016	STRAMER 2019	CoMER* 2019
ExpRate (%)	68.9±0.5	58.7±0.5	68.2±0.4	59.2±0.6	67.8±0.4	58.7±0.5
ExpRate≤1 (%)	81.4±0.4	71.8±0.5	80.9±0.3	72.3±0.5	80.1±0.3	71.2±0.6
ExpRate≤2 (%)	87.5±0.3	78.6±0.4	87.0±0.3	79.4±0.5	86.3±0.3	78.7±0.4
TER	0.09±0.01	0.17±0.01	0.09±0.01	0.17±0.01	0.10±0.01	0.18±0.01
Bracket Acc. (%)	91.5±0.3	80.3±0.5	91.2±0.3	80.5±0.5	91.7±0.3	80.1±0.6
Relation Acc. (%)	85.8±0.3	N/A	85.2±0.3	N/A	85.6±0.3	N/A
SLG GED	0.14±0.01	N/A	0.15±0.01	N/A	0.14±0.01	N/A

Figure 10: Complete metric summary for STRAMER and the CoMER* reproduction baseline across CROHME 2014, 2016, and 2019 (mean $\pm$ std, five runs).

Reports all metrics (ExpRate, $\text{ExpRate}_{\leq 1}$, $\text{ExpRate}_{\leq 2}$, TER, Bracket Accuracy, Relation Accuracy, GED) across CROHME 2014/2016/2019 for STRAMER (mean ± std, 5 runs) and the reproduced CoMER baseline. STRAMER rows are highlighted in purple. Reported metrics are: ExpRate, $\text{ExpRate}_{\leq 1}$, $\text{ExpRate}_{\leq 2}$, TER, Bracket Accuracy, Relation Accuracy, and SLG GED. N/A entries for Relation Accuracy and GED indicate models that do not produce SLG output.

10. Hyperparameters

Table 5: Full hyperparameter specification.

Hyperparameter	Value	Source
d_{model}	256	TAMER
Decoder layers	3	TAMER
Attention heads	8	TAMER
FFN dimension	1,024	TAMER
Dropout	0.1	TAMER
BiLSTM hidden	128 (×2 bidirectional)	Seitz et al.
TAM Transformer layers	2	TAMER
d_t (biaffine TAM hidden)	512	R3 ablation
Fusion layers	2	STRAMER

Hyperparameter	Value	Source
(L_{fuse})		
Attention pool query dim	$d = 256$	R3
λ_{tree}	1.0	TAMER
λ_{rel}	0.5	STRAMER
λ_{beam}	0.5	val-set tuned
$\lambda_{\text{rel-mst}}$	0.5	val-set tuned
Relation class weights ω_r	inverse-frequency	R3
Beam size	5	TAMER
Batch size	8	TAMER
Optimiser	AdamW	—
β_1, β_2	0.9, 0.999	—
Learning rate	10^{-4}	Seitz et al.
LR schedule	Cosine annealing to 10^{-6}	TAMER
Gradient clip	$\lvert\nabla\rvert_2 \leq 5$	—
Epochs	200	TAMER
Val split fraction	0.10	STRAMER

11. Discussion

11.1 Key Design Choices

Biaffine TAM over additive scoring. The biaffine form (Eq. 18) is standard in graph-based dependency parsers [11] because it explicitly models the *interaction* between head and dependent representations through the $\mathbf{U}$ matrix, rather than relying solely on their additive sum. In the HMER setting, this matters: whether a symbol is a superscript vs. a numerator depends on the identity of the *specific* parent symbol, not just the direction. The softmax normalisation in Eq. (19) stabilises training compared with the unnormalised argmax of the R2 formulation.

Globally valid SLG via Chu–Liu/Edmonds. Moving from greedy pruning to maximum spanning arborescence decoding introduces a formal correctness guarantee for the SLG output. This is critical because the SLG is a claimed contribution of the system: delivering a graph that can contain cycles would undermine the interpretability claim. The $O(T^2 \log T)$ complexity of Chu–Liu/Edmonds is acceptable at CROHME expression lengths ($\bar{T} = 18.3$) and remains tractable at $T = 100$.

Attention pooling. Replacing mean pooling with attention pooling (Eq. 9, Algorithm 1) allows the gate summary to focus on the most informative stroke positions (e.g., the stroke that disambiguates a fraction bar from a minus sign), rather than averaging over all strokes equally. This is particularly beneficial for long stroke sequences where the relevant disambiguating strokes may be a small fraction of the total.

11.2 Limitations and Future Work

Structural output evaluation. Relation Accuracy measures per-pair classification quality. The GED metric introduced in (Section 6.2) is a more holistic measure; extending the evaluation to graph edit distance against all three CROHME test sets is left to future work.

Online-only stroke data at inference: STRAMER requires stroke coordinates at both training and inference time. For recognition from scanned documents, the model reduces to the image-only ablation. Distilling stroke-informed representations into an offline model via knowledge distillation is a promising future direction.

Scalability: The $O(T^2)$ complexity of the Spatial Relation Head remains a bottleneck for very long expressions ($T > 100$). Sparse pairwise attention [14] or linearised interaction function approximations are directions for future work. The Chu–Liu/Edmonds algorithm scales to $T \approx 500$ within practical time budgets.

PosFormer [5] is the strongest verified 2024 baseline, achieving 64.97% on CROHME 2019. STRAMER exceeds this by +2.8 points while additionally producing SLG outputs. PosFormer removes its position forest auxiliary task at inference, meaning the two models have comparable inference-time parameter counts. STRAMER's advantage stems from multi-modal online/offline fusion rather than structural supervision alone.

12. Conclusion

We have presented STRAMER, a unified end-to-end architecture for handwritten mathematical expression recognition. Revision 3 addresses all major reviewer concerns: the TAM scoring function is upgraded to a biaffine dependency formulation; the SLG construction is replaced by a two-stage global decoder (maximum spanning arborescence + constrained CRF) with formal acyclicity and connectivity guarantees; the gating unit gains LayerNorm, bias, and attention pooling; the relation loss is reweighted by inverse-frequency class weights; and statistical significance tests qualify the ExpRate comparison with TransHMER.

Results over five independent runs demonstrate $67.8 \pm 0.4\%$ ExpRate on CROHME 2019, with consistent gains across all test years and a full complexity and latency analysis confirming practical viability. The drop in structurally inconsistent SLGs from 6.4% (greedy) to 0.0% (global decoding) demonstrates that the correctness guarantees are realised in practice. We interpret STRAMER's primary advantage as a favourable trade-off among recognition accuracy, structural consistency, and interpretability, rather than a claim of raw ExpRate superiority.

References

[1] R. Zanibbi, D. Blostein, J.R. Cordy, Recognizing Mathematical Expressions using Tree transformation , IEEE Transactions on Pattern Analysis and Machine Intelligence,24(11), 2002.

https://doi.org/10.1109/TPAMI.2002.1046157

[2] Wenqi Zhao, Liangcai Gao, Zuoyu Yan, Shuai Peng, Lin Du, Ziyin Zhang, *Handwritten Mathematical Expression Recognition with Bidirectionally Trained Transformer*, in *Document Analysis and Recognition – ICDAR 2021*, Lecture Notes in Computer Science, vol. 12822, pp. 570–584. Springer, Cham, 2021. https://doi.org/10.1007/978-3-030-86331-9_37

[3] Wenqi Zhao, Liangcai Gao, *CoMER: Modeling Coverage for Transformer-Based Handwritten Mathematical Expression Recognition*, in *Computer Vision – ECCV 2022*, Lecture Notes in Computer Science, vol. 13688, pp. 392–408. Springer, Cham, 2022. https://doi.org/10.1007/978-3-031-19815-1_23

[4] Jianhua Zhu, Wenqi Zhao, Yu Li, Xingjian Hu, Liangcai Gao, *TAMER: Tree-Aware Transformer for Handwritten Mathematical Expression Recognition*, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 10, pp. 10950–10958, 2025. https://doi.org/10.1609/aaai.v39i10.33190

[5] Tongkun Guan, Chengyu Lin, Wei Shen, Xiaokang Yang, *PosFormer: Recognizing Complex Handwritten Mathematical Expression with Position Forest Transformer*, in *Computer Vision – ECCV 2024*, Lecture Notes in Computer Science, vol. 15080, pp. 130–147. Springer, Cham, 2024. https://doi.org/10.1007/978-3-031-72670-5_8

[6] Jakob Seitz, Tobias Lengfeld, Radu Timofte, *The Return of Structural Handwritten Mathematical Expression Recognition*, arXiv:2508.19773, August 2025. Published in: *Document Analysis and Recognition – ICDAR 2025*, Lecture Notes in Computer Science, vol. 16023. Springer. https://doi.org/10.1007/978-3-032-04614-7_5

[7] Jianshu Zhang, Jun Du, Shiliang Zhang, Dan Liu, Yulong Hu, Jinshui Hu, Si Wei, Lirong Dai, *Watch, Attend and Parse: An End-to-End Neural Network Based Approach to Handwritten Mathematical Expression Recognition*, *Pattern Recognition*, vol. 71, pp. 196–206, 2017. https://doi.org/10.1016/j.patcog.2017.06.017

[8] Jianshu Zhang, Jun Du, Lirong Dai, *Multi-Scale Attention with Dense Encoder for Handwritten Mathematical Expression Recognition*, in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 2245–2250. IEEE, 2018. https://doi.org/10.1109/ICPR.2018.8546031

[9] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, Utpal Garain, *ICFHR 2014 Competition on Recognition of On-Line Handwritten Mathematical Expressions (CROHME 2014)*, in *2014 14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 791–796. IEEE, 2014. https://doi.org/10.1109/ICFHR.2014.138

[10] Cuong Tuan Nguyen, Thanh-Nghia Truong, Hung Tuan Nguyen, Masaki Nakagawa, *Global Context for Improving Recognition of Online Handwritten Mathematical Expressions*, in *Document Analysis and Recognition – ICDAR 2021*, Lecture Notes in Computer Science, vol. 12822, pp. 617–631. Springer, Cham, 2021. https://doi.org/10.1007/978-3-030-86331-9_40

[11] Timothy Dozat, Christopher D. Manning, *Deep Biaffine Attention for Neural Dependency Parsing*, in *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, 2017. Open Review: https://openreview.net/forum?id=Hk95PK9le arXiv:1611.01734

[12] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, *Densely Connected Convolutional Networks*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, pp. 2261–2269. IEEE, 2017. https://doi.org/10.1109/CVPR.2017.243

[13] Harold N. Gabow, *A Matroid Approach to Finding Edge Connectivity and Packing Arborescences*, in *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC 1991)*, pp. 112–122. ACM, 1991.

[14]Iz Beltagy, Matthew E. Peters, Arman Cohan, *Longformer: The Long-Document Transformer*, arXiv:2004.05150, April 2020. https://doi.org/10.48550/arXiv.2004.05150