A Tree-Based Structure-Aware Transformer Decoder for Image-To-Markup Generation

Shuhan Zhong
The Hong Kong University of Science and Technology
Hong Kong SAR, China
szhongaj@cse.ust.hk

Sizhe Song
The Hong Kong University of Science and Technology
Hong Kong SAR, China
ssongad@cse.ust.hk

Guanyao Li*
The Hong Kong University of Science and Technology
Hong Kong SAR, China
gliaw@cse.ust.hk

S.-H. Gary Chan
The Hong Kong University of Science and Technology
Hong Kong SAR, China
gchan@cse.ust.hk

ABSTRACT

Image-to-markup generation aims at translating an image into markup (structured language) that represents both the contents and the structural semantics corresponding to the image. Recent encoder-decoder based approaches typically employ string decoders to model the string representation of the target markup, which cannot effectively capture the rich embedded structural information. In this paper, we propose TSDNet, a novel **T**ree-based **S**tructure-aware Transformer **D**ecoder **Net**work to directly generate the tree representation of the target markup in a structure-aware manner. Specifically, our model learns to sequentially predict the node attributes, edge attributes, and node connectivities by multi-task learning. Meanwhile, we introduce a novel tree-structured attention to our decoder such that it can directly operate on the partial tree generated in each step to fully exploit the structural information. TSDNet doesn't rely on any prior assumptions on the target tree structure, and can be jointly optimized with encoders in an end-to-end fashion. We evaluate the performance of our model on public image-to-markup generation datasets, and demonstrate its ability to learn the complicated correlation from the structural information in the target markup with significant improvement over state-of-the-art methods by up to 5.6% in mathematical expression recognition and up to 35.34% in chemical formula recognition.

CCS CONCEPTS

• **Computing methodologies** → **Object recognition**; *Natural language generation*; Multi-task learning.

KEYWORDS

Image-To-Markup Generation, Tree-Structured Attention, Tree Decoder, Tree Generation

ACM Reference Format:
Shuhan Zhong, Sizhe Song, Guanyao Li, and S.-H. Gary Chan. 2022. A Tree-Based Structure-Aware Transformer Decoder for Image-To-Markup Generation. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), Oct. 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3503161.3548424

1 INTRODUCTION

Image-to-markup generation is to translate an image into markup (structured language) that represents both the contents and the structural semantics corresponding to the image [9]. Typical image-to-markup generation tasks include mathematical expression recognition [24] that translates images into LaTeX [15], chemical formula recognition [28] that translate images into the chemical language SMILES [38]. It may even be extended to automatically generate programming code from images of graphical user interfaces [4, 6]. Research on image-to-markup generation dates back to as early as 1960s [2], and is gaining increasing attention due to its importance in education, office automation, and multimedia data mining. Markup languages are naturally tree-structured to express not only the contents but also the hierarchies and relationships between the contents [46], such as sub- and super-scripts in mathematical expressions, as well as bonds and branches in chemical formulas (Figure 1). It is hence essential to consider both the contents and their structural semantics in image-to-markup generation.

Traditional approaches process structural analysis separately from the content recognition by handcrafted grammar rules, which is labor-intensive and inflexible [1, 3, 12, 17, 23]. Recent approaches instead employ deep learning models to process the image-to-markup generation in an end-to-end manner [18, 39, 40]. Specifically, they rely on string decoders such as recurrent neural networks (RNNs) and Transformer [34] to learn to predict the markup language strings as output. However, in markup language strings, the structural information is usually described by additional formatting tokens (e.g., "_", "^", "{", "}" in LaTeX strings, and "(", ")", "=" in SMILES, as shown in Figure 1). So, it is hard for those string decoders, which are originally designed for linear sequences, to consider the rich structural relationships embedded in the markup. Because of this,

*Also with Guangzhou Urban Planning and Design Survey Research Institute, Guangdong Enterprise Key Laboratory for Urban Sensing, Monitoring and Early Warning.

MM '22, October 10–14, 2022, Lisboa, Portugal.

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548424

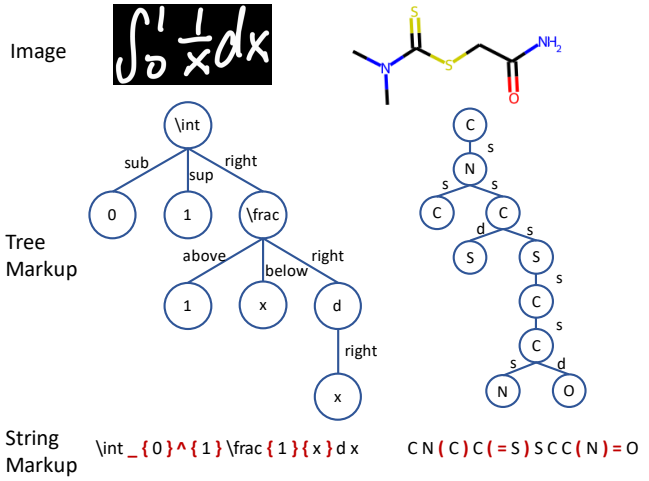


Figure 1: Examples of image-to-markup generation.

those string-based methods struggle to generalize when the structural complexity of markup increases [46]. In order to consider the structural information, Zhang et al. [46] propose a tree-structured decoder to directly generate the tree representation of the markup. While impressive, their method only considers parent-child correlation, while other structural correlations between the tree nodes, such as siblings, grandparent-grandchild remain unexploited.

In this work, based on the insight of modeling the target markup as tree, we propose to fully exploit the structural semantics by building a tree-based and structure-aware decoder model to generate the target tree. This task is challenging because of the following. Firstly, we need to predict three types of essential attributes for the tree-based generation, namely node attributes, edge attributes, and node connectivities. These attributes are highly coupled, and how to integrate them into the model is non-trivial. Secondly, we want our model to be structure-aware such that it can fully leverage the partial tree generated by previous steps to generate the next step. However, off-the-shelf structure-aware models such as Graph Attention Networks (GATs) [35] are generally designed for the encoding scenario, thus not readily extensible to our tree decoding scenario where the partial tree needs to be incrementally modeled during the decoding process.

To overcome the challenges mentioned above, we propose TSDNet, a novel **T**ree-based **S**tructure-aware Transformer **D**ecoder **Net**work. TSDNet is well-designed to fully exploit the tree structure in the target markup that: (i) it learns to explicitly generate the tree representation of the target markup, without any prior assumptions on the tree structure (e.g. maximum tree depth or branching factor), (ii) it directly operates on the target tree structure, to inherently leverage the structural information during the decoding process. Specifically, TSDNet is designed based on the Transformer decoder to model the tree generation process in the depth-first search (DFS) order. It relies on multi-task learning to jointly model and predict the node attributes, edge attributes, and node connectivities. Meanwhile, we introduce a novel tree-structured attention to replace the feed-forward sublayer in the original Transformer decoder, enabling it to directly operate on the partial tree and adaptively learn to capture complicated correlations in the tree such as parent-child, grandparent-grandchild and siblings. Furthermore, TSDNet takes the advantages of the attention mechanism that the sequential process can be parallelly computed during training. Besides, it can be jointly optimized with the encoder in the end-to-end fashion.

To summarize, we make the following contributions in this paper:

- *A tree-structured attention mechanism:* We propose a novel tree-structured attention mechanism to inherently leverage the structural correlations in partial trees under the sequential tree decoding scenario. It can directly operate on the partial tree during the decoding process, and can adaptively learn to capture complicated correlations in the tree such as parent-child, grandparent-grandchild and siblings within each layer.
- *A tree-based structure-aware Transformer decoder network:* Based on our tree-structured attention mecheanism, we propose TSDNet, a novel tree-based structure-aware Transformer decoder network to directly generate the tree representation of the target markup. TSDNet doesn't rely on any prior assumptions on the structure of the target tree, and can be jointly and efficiently optimized with general encoders by end-to-end training.
- *Extensive experiments to validate the effectiveness of TSDNet:* We conduct extensive experiments to evaluate the performance of TSDNet on public image-to-markup generation datasets, and demonstrate significant improvements over state-of-the-art methods (by up to 5.6% in mathematical expression recognition tasks and up to 35.34% in the chemical formula recognition task in terms of ExpRate). We also conduct experiments to show the effectiveness of TSDNet to learn the complicated correlation from the structural information in the target markup.

The remainder of this paper is organized as follows. We present related works in Secion 2, followed by preliminaries in Section 3. We then introduce our tree-structured attention mechanism in Section 4, and the TSDNet in Section 5. We illustrate the experimental settings and results in Section 6, and conclude in Section 7.

2 RELATED WORKS

2.1 Image-to-Markup Generation

Unlike typical object recognition problem that recognizes contents from images, image-to-markup generation requires to further analyze the structural relationships between the recognized contents. Early approaches for image-to-markup generation process the recognition and structural analysis separately and use handcrafted grammars to handle the structural analysis [1, 3, 12, 17, 23], which requires a large amount of manual work to develop the grammar rules.

Inspired by the neural encoder-decoder model developed for image captioning [13, 42], recent studies essentially formulate image-to-markup generation as an image-to-sequence problem. These models generally employ convolutional neural networks (CNNs) as encoder and use string decoders like RNN or Transformer [34] to generate the markup strings in an end-to-end manner [9, 18, 39, 40, 47]. However, these string decoders are originally

designed for linear sequences and have no inductive bias to leverage the rich structural semantics embedded in the markup strings. As a result, these string-based methods fail to generalize as the structural complexity of the target markup increases [46]. To alleviate this problem, Zhang et al. [46] propose to directly model the tree representation of the markup, by decomposing it into a sequence of parent-child pairs and generating it with an RNN-based model. While impressive, their method is still not fully structure-aware that it only considers parent-child correlation, other structural correlations between tree nodes, such as siblings and grandparent-grandchild remain unexploited.

Compared with existing string-based methods, our proposed TSDNet is tree-based that it directly models the target tree markup. Compared with the tree-based method by Zhang et al. [46], TSDNet is fully structure-aware that it relies on a novel tree-structured attention to directly operate on the tree structure, and inherently leverage the structural correlations including parent-child, grandparent-grandchild, and siblings.

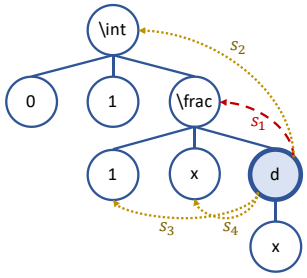
2.2 Tree-Structured Neural Networks

There have been a multitude of works that showed gains from incorporating tree structures into neural models in other research strands such as natural language processing [8, 27, 30, 33, 36, 37] and representation learning of programming code [10, 31, 32, 41, 45, 48]. However, the majority of them focus on incorporating structural information into encoders to help generate more meaningful representations, and much fewer works try to address the decoding of structural information. Different from tree-structured encoders that can take in the entire tree as input, tree-structured decoders must dynamically model the already generated partial tree throughout the generation process [31], which brings additional conceptual and technical complexity into the problem [8]. Among existing works on tree decoding, some rely on task-specific knowledge to decode the target tree [8, 32, 45], which is non-trivial to be adapted to other tasks. Others make prior assumptions on the target tree structure, such as fixing the branching factor[7, 22, 36], and the maximum tree depth [31] to simplify the generation process, which also places restrictions on their applications. Compared with the above mentioned tree-structured neural networks, our proposed TSDNet works without any prior assumption on the target tree structure.

3 PRELIMINARIES

Definition: Image-to-Markup Generation. The image-to-markup generation problem is defined as converting a source image $\mathcal{X}$ into the target markup $\mathcal{M}$ that fully describes both the contents and the structural semantics in $\mathcal{X}$. An image-to-markup system learns to generate $\mathcal{M}$ by maximizing the conditional probability $P(\mathcal{M}|\mathcal{X})$.

Definition: Tree. A tree can be viewed as a special type of directed acyclic graph, where each edge directs from the parent node to the child node, and each node except for the root node has only one incoming edge from its parent node (referred to as parent edge). For a tree $\mathcal{T}$ with n nodes, we denote the tree nodes by their order in the DFS traversal of the tree as $\{1, \ldots, n\}$. Then the tree can be represented by three sequences with length n as: $\mathcal{T} = (v_{1:n}, e_{1:n}, p_{1:n})$, where for node i, v_i denotes its node attribute,



	\int	0	1	\frac	1	x	d	x
GAT	$-\infty$	$-\infty$	$-\infty$	s_1	$-\infty$	$-\infty$	$-\infty$	$-\infty$
TSA	$s_2+\beta_2$	$-\infty$	$-\infty$	$s_1+\beta_1$	$s_3+\beta_2$	$s_4+\beta_2$	$-\infty$	$-\infty$

Figure 2: Attention scores for node "d" under GAT and TSA with D_{max}=2.

e_i denotes the attribute of its parent edge, and $p_i = j$ denotes that its parent is node j. e_1 and p_1 are dummy here since the root node 1 has no parent. Meanwhile, under DFS, a parent node is always traversed before its children so we always have $1 \leq p_i < i$. We further denote the attribute of node i's parent node as v_i^p where $v_i^p = v_{p_i}$.

Problem Formulation: Generating Markup as Tree. We assume the target markup $\mathcal{M}$ can be represented as a tree $\mathcal{T} = (v_{1:n}, e_{1:n}, p_{1:n})$. Thus we can generate the target markup in tree representation $\mathcal{T}$ by sequentially predicting the node attribute v_i and edge attribute e_i, and identifying its parent node p_i from the previously generated nodes. The generation process can be factorized as:

$$\max_{\mathcal{T}} P(\mathcal{T}|\mathcal{X}) = \prod_{i=1}^{n} \max_{v_i, e_i, p_i} P(v_i, e_i, p_i | v_{<i}, e_{<i}, p_{<i}, \mathcal{X}) \quad (1)$$

where $(v_{<i}, e_{<i}, p_{<i})$ stands for the already generated partial tree.

4 TREE-STRUCTURED ATTENTION

We adopt the idea of the GAT [35] to introduce tree structure into the canonical Transformer decoder. The vanilla GAT computes in each layer the attention score of each node over all its direct graph neighbor nodes, and relies on stacking multiple layers to increase the receptive field of the network. However, as we generate the tree nodes sequentially, each node should only be allowed to compute attention with earlier positions in the sequence. But under the DFS order, only the parent node appears in earlier positions in the sequence, and child nodes always appear in later positions. Since a non-root tree node has only one parent node, if we directly apply GAT to our model, each node will only be able to depend on its one-and-only parent node for feature extraction. This severely limits the ability of the vanilla graph attention mechanism when applied to our tree decoder. To cope with it, we revise the graph attention computation by enabling attention between tree nodes that are not neighbors, and update the attention score by adding a learnable bias term according to the distance in the tree (Figure 2).

Given a tree with t nodes, we first compute the pairwise attention score $\boldsymbol{S} = \{s_{ij}\} \in \mathbb{R}^{t\times t}$ between nodes:

$$s_{ij} = \text{LeakyReLU}(\boldsymbol{a}^T \cdot [\boldsymbol{W}\boldsymbol{h_i} \parallel \boldsymbol{W}\boldsymbol{h_j}]), \tag{2}$$

where $\boldsymbol{a}$ and $\boldsymbol{W}$ are learnable parameters, $\parallel$ denotes the concatenation operation, and $\boldsymbol{H} = \{\boldsymbol{h_i}\} \in \mathbb{R}^{t\times d}$ denotes the input representation vectors of tree nodes. Then, we consider the tree edges as undirected and denote its adjacency matrix as $\boldsymbol{A}$. Next, we define the n-hop adjacency matrix as $\boldsymbol{A^{(n)}} = \{a_{ij}^{(n)}\} \in \{0,1\}^{t\times t}$, where $a_{ij}^{(n)} = 1$ means node i and j are n-hop neighbor in the tree. It is obvious that $\boldsymbol{A^{(1)}} = \boldsymbol{A}$. We can further compute $\boldsymbol{A^{(n)}}$ for $n \geq 2$ according to the graph diffusion process:

$$\boldsymbol{A^{(n)}} = \boldsymbol{A^{(n-1)}}\boldsymbol{A^{(1)}}. \tag{3}$$

Next, the attention bias is computed as the weighted sum of all n-hop adjacency matrices:

$$\boldsymbol{B} = \{b_{ij}\} = \sum_{i=1}^{n} \beta_i * \boldsymbol{A^{(i)}}, \tag{4}$$

where β_i are learnable parameters. We update the attention scores $\boldsymbol{S}$ by adding $\boldsymbol{B}$ to it, and mask out those node pairs that are not connected within n-hops to obtain the new attention scores $\boldsymbol{S'} = \{s'_{ij}\}$ where

$$s'_{ij} = \begin{cases} s_{ij} + b_{ij}, & \text{node } i, j \text{ are connected within } n\text{-hops}, \\ -\infty, & \text{otherwise.} \end{cases} \tag{5}$$

The new attention scores are further normalized by softmax and used as weights to generate the new node representations $\boldsymbol{H'}$ following the operations in the original GAT [35]:

$$\boldsymbol{H'} = \sigma(\text{Softmax}(\boldsymbol{S'})\boldsymbol{W}\boldsymbol{H}). \tag{6}$$

We treat the max hop number considered in each TSA layer D_{max} as a hyper parameter. It should be noted that when $D_{max} = 1$, TSA works similarly to the vanilla GAT. But when D_{max}>1 , our TSA allows information aggregation based on distances in the tree, which enables the model to adaptively learn different correlations on the tree such as parent-child, grandparent-grandchild, and siblings. We summarize the computation from Equations 2 to 6 as Tree-Structured Attention (TSA) and denote it as:

$$\boldsymbol{H'} = \text{TSA}(\boldsymbol{H}, \boldsymbol{A}). \tag{7}$$

5 TSDNET

Figure 3 shows the overall architecture of the proposed TSDNet. TSDNet works with an encoder to extract visual features from the input image. In each generation step, the partial tree generated in previous steps is embedded by the Tree Embedding Block, and then TSDNet relies on the Child Prediction Block and the Parent Prediction Block to operate on the partial tree and incorporate visual features to generate the vector representations of the next child node and its parent node. These vector representations are later used to predict node attribute, and further fused in the Edge Prediction Block and the Position Prediction Block to predict edge attribute and node connectivity. In this section, we elaborate the functionality and design of each block, and give the training and inference details of the TSDNet.

5.1 Encoder

In this work we employ CNN as the encoder. The CNN encoder takes the raw image as input and produces a feature map $\boldsymbol{X} \in \mathbb{R}^{h\times w\times d}$, where h and w are the spatial dimensions, and d is the feature dimension of the resulted feature map. We follow Carion et al. [5] to obtain the 2d sinusoidal positional encoding by computing and concatenating sine and cosine functions with different frequencies for height and width coordinates respectively. We add the positional encoding to the feature map, and then flatten the spatial dimensions to get a sequence of vectors, denoted as $\boldsymbol{X^{in}} \in \mathbb{R}^{hw\times d}$.

5.2 Tree Embedding Block

Given a partial tree with t nodes specified as $(v_{1:t}, e_{1:t}, p_{1:t})$, we use look-up tables to embed node and edge attributes as real-valued vectors respectively, denoted as $\{\boldsymbol{v_1}, \ldots, \boldsymbol{v_t}\}$ and $\{\boldsymbol{e_1}, \ldots, \boldsymbol{e_t}\}$. We then concatenate the vector embedding of each node and their parent edge, and use a linear transformation to fuse these information into one vector space:

$$\boldsymbol{h_i} = \boldsymbol{W_{ve}}[\boldsymbol{v_i} \parallel \boldsymbol{e_i}]. \tag{8}$$

Sinusoidal positional encoding [34] is computed and added to $\boldsymbol{h_i}$. Meanwhile, we treat the tree edges as undirected and compute the adjacency matrix of the tree $\boldsymbol{A_t} = \{a_{ij}\} \in \{0,1\}^{t\times t}$ from $\{p_i\}$ by:

$$a_{ij} = \begin{cases} 1, p_i = j \text{ or } p_j = i, \\ 0, \text{otherwise.} \end{cases} \tag{9}$$

By doing this, the partial tree is represented as a sequence of feature vector $\boldsymbol{H_t^{in}} = \{\boldsymbol{h_i}\} \in \mathbb{R}^{t\times d}$ and its adjacency matrix $\boldsymbol{A_t}$.

5.3 Child Prediction Block

The Child Prediction Block takes as input the image feature sequence $\boldsymbol{X^{in}}$ from the encoder as well as the partial tree feature sequence $\boldsymbol{H_t^{in}}$ and its adjacency matrix $\boldsymbol{A_t}$ from the tree embedding block. Based on these inputs, the Child Prediction Block learns to operate on the partial tree structure specified by $\boldsymbol{A_t}$ and extract features from $\boldsymbol{X^{in}}$ and $\boldsymbol{H_t^{in}}$ to generate the hidden representation of the next child node. The Child Prediction Block is composed of a stack of N_c tree decoder layers. Inspired by Li et al. [19] and Yang et al. [44], we build each tree decoder layer by replacing the fully-connected sublayer in the vanilla Transformer decoder with TSA (Figure 4). The advantage of this design is that it enjoys the capabilities of both self-attention to capture global pairwise dependencies and tree-structured attention to capture tree structure information. We denote the scaled dot-product attention in the tree decoder layer as:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{10}$$

where $\frac{1}{\sqrt{d_k}}$ is the scaling factor. Within each tree decoder layer l, the first sublayer is a multi-head self-attention that operates over the hidden representations from previous decoder layer $\boldsymbol{H^{(l-1)}}$:

$$\boldsymbol{H_1^{(l)}} = \text{LayerNorm}(\text{Attention}(\boldsymbol{Q_H}, \boldsymbol{K_H}, \boldsymbol{V_H}) + \boldsymbol{H^{(l-1)}}), \tag{11}$$

where $\boldsymbol{Q_H}, \boldsymbol{K_H}, \boldsymbol{V_H}$ are the query, key, and value vectors transformed from $\boldsymbol{H^{(l-1)}}$. The second sublayer performs attention over

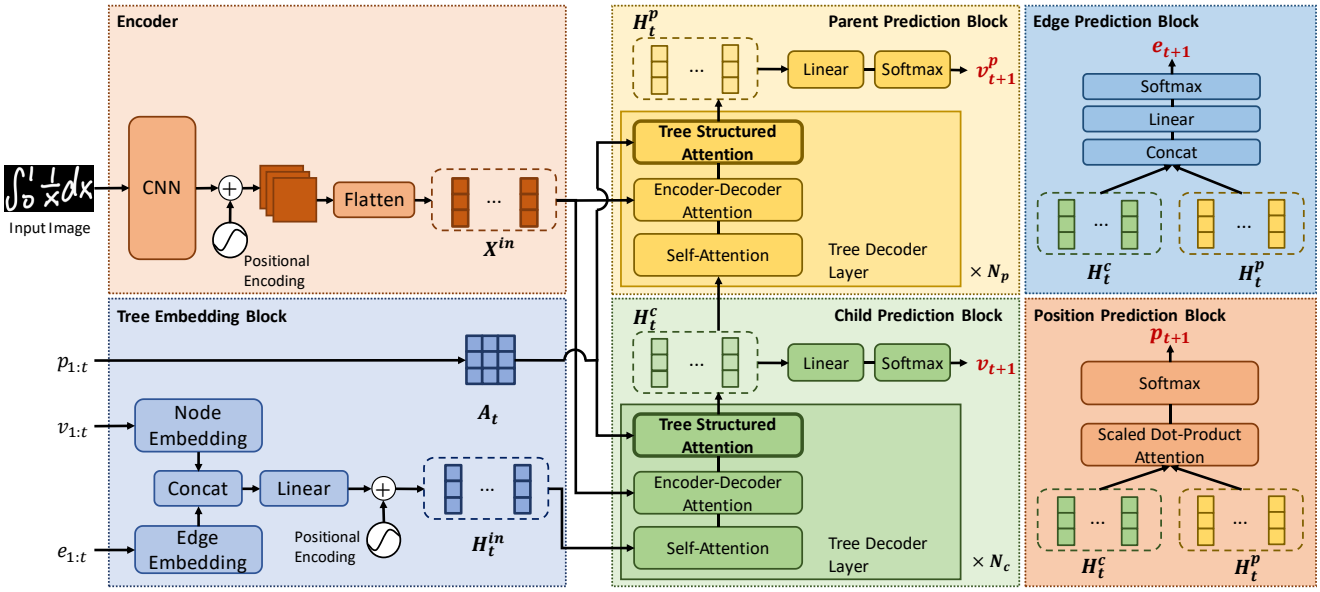


Figure 3: Overall architecture of TSDNet.

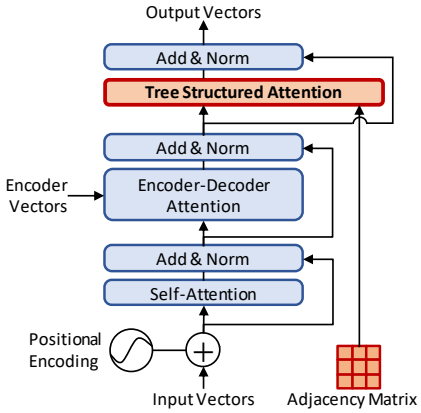


Figure 4: Tree decoder layer.

the output of the encoder representation:

$$H_2^{(l)} = \text{LayerNorm}(\text{Attention}(Q_{H_1}, K_X, V_X) + H_1^{(l)}), \quad (12)$$

where Q_{H_1} is transformed from $H_1^{(l)}$, K_X and V_X are transformed from the encoder representation X^{in}. The last sub-layer performs tree-structured attention over the hidden representations from previous sublayer:

$$H^{(l)} = \text{LayerNorm}(\text{TSA}(H_2^{(l)}, A_t) + H_2^{(l)}). \quad (13)$$

We denote the output of the last tree decoder layer in the Child Prediction Block as $H_t^c = \{h_i^c\}$. We predict the next node attribute v_{t+1} from h_t^c:

$$P(\hat{v}_{t+1}|v_{1:t}, e_{1:t}, p_{1:t}, \mathcal{X}) = \text{Softmax}(W_c h_t^c), \quad (14)$$

where W_c is learnable parameter.

5.4 Parent Prediction Block

The Parent Prediction Block is another stack of N_p tree decoder layers. It takes as input the image feature sequence X^{in}, the output feature sequence from the Child Prediction Block H_t^c and the adjacency matrix A_t. Based on these inputs, the Parent Prediction Block learns to operate on the partial tree structure specified by A_t and extract features from X^{in} and H_t^c to generate the hidden representations of the next child node's parent node. We denote the output of the last tree decoder layer in the Parent Prediction Block as $H_t^p = \{h_i^p\}$. We predict the next parent node's attribute v_{t+1}^p from h_t^p:

$$P(\hat{v}_{t+1}^p|v_{1:t}, e_{1:t}, p_{1:t}, \mathcal{X}) = \text{Softmax}(W_p h_t^p), \quad (15)$$

where W_p is learnable parameter.

5.5 Edge Prediction Block

In the Edge Prediction Block we combine the output of the Child and Parent Prediction Block to predict the attribute of the next node's parent edge:

$$P(\hat{e}_{t+1}|v_{1:t}, e_{1:t}, p_{1:t}, \mathcal{X}) = \text{Softmax}(W_e[h_t^c \parallel h_t^p]), \quad (16)$$

where W_e is learnable parameter.

5.6 Position Prediction Block

In the Position Prediction Block we try to identify which previous generated node in the partial tree is the parent node of the next node. As we have the representation vector of the previous generated nodes as well as the next parent node, we adopt the idea of the pointer network [29] and use scaled dot-product attention [34] to compute the probability of previous generated nodes being the next parent node. Specifically, we use h_t^p as the query and H_t^c as keys, and the attention scores are normalized by softmax to compute the

probability:

$$\boldsymbol{q_t} = \boldsymbol{W_q h_t^p}, \boldsymbol{K} = \boldsymbol{W_K H_t^c}, \tag{17}$$

$$P(\hat{p}_{t+1}|v_{1:t}, e_{1:t}, p_{1:t}, \mathcal{X}) = \text{Softmax}(\frac{q_t^T K}{\sqrt{d}}), \tag{18}$$

where $\boldsymbol{W_q}$ and $\boldsymbol{W_K}$ are learnable parameters, and $\frac{1}{\sqrt{d}}$ is the scaling factor.

5.7 Training and Inference

We train TSDNet by simultaneously optimizing multiple training losses from the prediction tasks. The loss function is:

$$\mathbb{L} = \lambda_1 \mathbb{L}_v + \lambda_2 \mathbb{L}_{v_p} + \lambda_3 \mathbb{L}_e + \lambda_4 \mathbb{L}_p, \tag{19}$$

where $\mathbb{L}_v, \mathbb{L}_{v_p}, \mathbb{L}_e, \mathbb{L}_p$ are cross-entropy losses computed from the child, parent, edge, and position predictions, respectively. During training we add special nodes before the root node and after the last child node of each tree to mark the start and end of the decoding. We follow the training process of the Transformer that we feed TSDNet with the right shifted sequences at once and let it predict the original sequences and compute the training loss. Subsequent masks are used to avoid information leakage from the future positions in attention computation in TSDNet.

During inference, we feed TSDNet with the start node, and let it loop until the end node is generated. Greedy search is employed for the generation. The node attributes, edge attributes, and node connectivities generated in the intermediate steps are added to the partial tree for future generation steps.

6 ILLUSTRATIVE EXPERIMENTAL RESULTS

In this section, we introduce the implementation details of TSDNet (Section 6.1), datasets for experiments (Section 6.2), baseline methods and evaluation metrics (Section 6.3). Then we compare the performance of TSDNet with state-of-the-art approaches (Section 6.4), perform ablation studies (Section 6.5), and evaluate the modeling ability and generalization ability in terms of the structural complexity of the target markup (Section 6.6). Next, we study effect of different D_{max} in TSA on the performance (Section 6.7), and finally study the effectiveness of TSDNet by visualization cases (Section 6.8).

6.1 Implementation Details

In this work we instantiate CNN in the encoder with DenseNet [11] following Zhang et al. [46] for fair comparison. For TSDNet, we set $N_c = N_p = 3$ and $D_{max} = 2$. The dimension for each tree decoder layer is set as 256, and 8 attention heads are computed. We optimize the model parameters with AdamW [21], and applied a cosine annealing learning rate scheduler [20], with the period to be 300 epochs, and peak learning rate to be 3×10^{-4} to adjust the learning rate during training. We further follow Vaswani et al. [34] to use a warmup training stage of 20 epochs. For the loss function, we use $\lambda_1 = \lambda_3 = \lambda_4 = 1$ and $\lambda_2 = 0.1$.

6.2 Datasets

We conduct experiments on two datasets, namely the *CROHME* dataset for handwritten mathematical expression recognition, and the *ZINC* dataset for chemical formula recognition:

The *CROHME* dataset is published by the Competition on Recognition of Handwritten Mathematical Expressions [24]. *CROHME* provides images containing handwritten mathematical expression and the corresponding LaTeXstrings as well as their tree representations as shown in Figure 1. We follow the experiment settings of Zhang et al. [46] to train the models on the *CROHME* training set and evaluate on the three test sets, namely *CRHOME14*, *CROHME16*, and *CROHME19*.

We construct another dataset *ZINC* for chemical formula recognition from the ZINC molecules dataset [14]. The ZINC molecules dataset provides 250K drug molecules in the SMILES format [38]. We randomly sample 39000 molecules from the ZINC molecules dataset, and use a widely-accepted software in chemistry named RDKit [16] to render the SMILES strings into images. We then convert the SMILES strings into trees as shown in Figure 1 according to the SMILES grammar. Finally we split these data into train/valid/test sets with 30000/3000/6000 samples for training and testing.

6.3 Baseline Methods and Evaluation Metrics

We compare the performance of our proposed TSDNet with state-of-the-art methods. For the *CROHME* test sets, we also provide results of competitive methods reported in the corresponding *CROHME* competitions for comparison.

- *UPV (for CROHME14 only) [25]:* It is a competitive grammar-based method reported in the CROHME 2014 competition.
- *WYGIWYS (for CROHME14 only) [9]:* It is a string-based method that uses CNN as encoder and RNN as decoder with a novel coarse-to-fine attention mechanism.
- *Tokyo (for CROHME16 only) [26]:* It is a competitive method reported in the CROHME 2016 competition. It uses CNN for symbol recognition and grammar-based method for structural analysis.
- *Univ. Linz (for CROHME19 only) [24]:* It is a competitive method reported in the CROHME 2019 competition. It uses Faster RCNN for symbol detection and Transformer to generate the target markup strings.
- *Transformer [34]:* It is a string-based method that employs DenseNet [11] as encoder and the Transformer decoder in a *small* configuration as decoder.
- *DenseWAP-TD [46]:* It is a tree-based method that employs DenseNet [11] as encoder and an RNN based decoder to learn to generate the tree representation of the target markup as a sequence of parent-child node pairs.
- *TSDNet-L:* It is a variant of our proposed TSDNet that the tree decoder layers are replaced with canonical Transformer decoder layers. It don't have TSA to operate on the tree and hence is not structure-aware.

We compare the model output with the ground truth by computing the edit distance between them. Then we evaluate the model performance using the following metrics:

- *ExpRate, ED1, and ED2:* Percentage of model outputs that have edit distance no greater than 0 (exact match), 1, and 2 to the ground truths.
- *StruRate (for tree-based model only):* Percentage of model outputs that have 0 edit distance to the ground truths when only node connectivity is considered.

Table 1: Comparison with state-of-the-art methods.

Dataset	Model	Decoder Type	ExpRate (%)	ED1 (%)	ED2 (%)	StruRate (%)
CROHME14	UPV	grammar-based	37.22	44.22	47.26	N/A
	WYGIWYS	string-based	40.40	56.10	59.90	N/A
	Transformer	string-based	48.17	59.63	63.29	N/A
	DenseWAP-TD (baseline)	tree-based	49.10	64.20	67.80	68.60
	TSDNet-L (ours)	tree-based	52.83 (3.73↑)	66.78 (2.58↑)	71.41 (3.61↑)	72.69 (4.09↑)
	TSDNet (ours)	tree-based	**54.70** (5.60↑)	**68.85** (4.65↑)	**74.48** (6.68↑)	**74.37** (5.77↑)
CROHME16	Tokyo	grammar-based	43.94	50.91	53.70	N/A
	Transformer	string-based	44.55	55.88	60.59	N/A
	DenseWAP-TD (baseline)	tree-based	48.50	62.30	65.30	65.90
	TSDNet-L (ours)	tree-based	51.18 (2.68↑)	68.00 (5.70↑)	71.67 (6.37↑)	70.14 (4.24↑)
	TSDNet (ours)	tree-based	**52.48** (3.98↑)	**68.26** (5.96↑)	**73.41** (8.11↑)	**71.88** (5.98↑)
CROHME19	Univ. Linz	string-based	41.49	54.13	58.88	N/A
	Transformer	string-based	44.95	56.13	60.47	N/A
	DenseWAP-TD (baseline)	tree-based	51.40	66.10	69.10	69.80
	TSDNet-L (ours)	tree-based	52.40 (1.00↑)	69.02 (2.92↑)	74.81 (5.71↑)	72.04 (2.24↑)
	TSDNet (ours)	tree-based	**56.34** (4.94↑)	**72.97** (6.87↑)	**77.84** (8.74↑)	**75.65** (5.85↑)
ZINC	Transformer	tree-based	47.30	79.22	91.47	N/A
	DenseWAP-TD (baseline)	tree-based	58.23	86.32	93.62	92.57
	TSDNet-L (ours)	tree-based	89.03 (30.80↑)	91.45 (5.13↑)	91.95	91.70
	TSDNet (ours)	tree-based	**93.57** (35.34↑)	**94.77** (8.45↑)	**95.28** (1.66↑)	**95.01** (2.44↑)

6.4 Comparison with State-of-the-art Methods

Table 1 summurizes the evaluation results of our model on four different test sets compared with baseline methods. As shown in the table, our proposed TSDNet significantly outperforms all state-of-the-art methods on all datasets in terms of all evaluation metrics, which demonstrates the success of TSDNet in modeling the image-to-markup generation process. In particular, all tree-based methods yield better results than string-based methods, which proves that tree-based decoders can better leverage the rich structural semantics in the target markup than string-based decoders. Among the tree-based methods, DenseWAP-TD only considers the parent-child correlation, and models the generation process by the recurrent mechanism, which has limited capability for long processes. Compared with DenseWAP-TD, TSDNet relies on the tree structured attention to learn different correlations on the tree, and models the generation process by attention mechanism, which has better capability for long and complex processes. These are the reasons why TSDNet outperforms DenseWAP-TD.

6.5 Ablation Study

In order to validate the efficacy of the overall TSDNet framework and the TSA mechanism we propose, we conduct experiments on *ZINC* with several variants of the TSDNet, where TSDNet-GAT replaces TSA in the TSDNet with vanilla GAT. The results are summarized in Table 2. Even without the TSA mechanism, the TSDNet-L shows a large ExpRate improvement by up to 41.77% compared with the canonical string-based Transformer decoder, which proves the efficacy of the overall TSDNet framework in modeling the target tree. We also observe that by using vanilla GAT with TSDNet, the ExpRate of TSDNet-GAT improves only 1.23% from TSDNet-L, but by extending vanilla GAT into TSA, the ExpRate of TSDNet improves 4.54% from TSDNet-L, which demonstrates the advantage of the proposed TSA mechanism over the vanilla GAT to model the structural correlations in the tree generation problem.

Table 2: Ablation study results on *ZINC*.

Model	Decoder Type	GAT	TSA	ExpRate (%)
Transformer	String-based	No	No	47.30
TSDNet-L	Tree-based	No	No	89.03
TSDNet-GAT	Tree-based	Yes	-	90.26
TSDNet	Tree-based	-	Yes	93.57

6.6 Structural Complexity

We conduct experiments on the *ZINC* dataset to evaluate the modeling ability and generalization ability of TSDNet in terms of the structural complexity of the target markup. To this end, we define the structural complexity of a tree as the number of leaf nodes in the tree, as a tree with only one leaf node can be viewed as a linear sequence from the root to the leaf, and more leaf nodes indicates the tree is less linear. Following this definition, we evenly split the *ZINC* dataset into three subsets, namely *Easy*, *Mid*, *Hard* according to the target tree complexity, each subset containing 13000 samples. We further randomly split each subset into train/valid/test sets with 10000/1000/2000 samples. Then we train and test our model and baseline models on these subsets as well as the *Full ZINC* dataset. The ExpRate is shown in Table 3.

From the results, we can observe that the TSDNet trained on *Full* performs more consistently on different structural complexity

Table 3: ExpRate (%) on different structural complexities.

Training Set	Model	Test Set			
		Easy	Mid	Hard	Full
Easy	Transformer	65.30	40.50	16.40	40.73
	DenseWAP-TD	72.80	49.30	24.55	48.88
	TSDNet (ours)	**89.50**	**75.65**	**43.65**	**69.60**
Mid	Transformer	53.85	47.30	30.55	43.90
	DenseWAP-TD	79.45	71.00	53.10	67.85
	TSDNet (ours)	**86.75**	**85.15**	**72.65**	**81.52**
Hard	Transformer	57.75	66.15	66.95	63.62
	DenseWAP-TD	66.95	53.85	37.70	52.83
	TSDNet (ours)	**81.55**	**86.35**	**84.25**	**84.05**
Full	Transformer	63.95	48.35	29.60	47.30
	DenseWAP-TD	75.35	59.75	39.60	58.23
	TSDNet (ours)	**95.50**	**94.80**	**90.40**	**93.57**

levels. Meanwhile, when tested on structural complexity levels unseen in the training set, TSDNet always generalizes better than other methods. It proves the effectiveness of our proposed TSDNet to learn and generalize the complex structural correlation in the target markup.

6.7 Hop Numbers in TSA

To further analyse how the TSA helps the model to be structure-aware, we perform experiments on the *ZINC* dataset by varying the max number of hops D_{max} in the TSA layers of TSDNet. As results shown in Figure 5, when $D_{max} = 1$, the TSA works like the vanilla GAT that suffers from very limited receptive field in each layer, and has poorer performance. Then we observe significant ExpRate improvement when D_{max} increases from 1 to 2, since information aggregation from multi-hop neighbors is enabled. The ExpRate doesn't change too much until D_{max} goes larger than 10, and then start to decrease slowly as D_{max} increases to 16. The reason is that we use the TSA together with self-attention to build each Tree Decoder Layer of TSDNet (Figure 4), so as to leverage the capability of self-attention to capture global pairwise dependencies. In this way, TSA works by introducing the tree locality into TSDNet for information exchange within a local connected region in the tree. Nevertheless, as D_{max} gets too large, TSA gradually loses the ability to provide the tree locality, and becomes more similar to self-attention that already exists in TSDNet, thus affecting the performance of the model.

6.8 Visualization

We visualize the encoder-decoder attention scores from different models for generation steps in *CROHME* test cases, and project them back onto the input images, as shown in Figure 6. We can observe from the figure that TSDNet always focuses on the image area containing the target to be decoded, while the attentions of Transformer and the TSDNet-L get more distracted to other image areas. It demonstrates that our proposed TSA module can help the model to better understand the structural correlation in the target tree and guide the model's attention on the image.

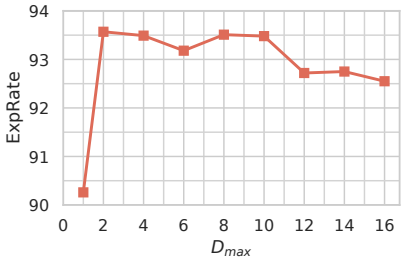


Figure 5: Performance of TSDNet with different D_{max} on *ZINC*.

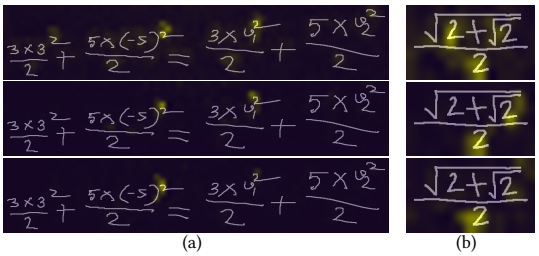


Figure 6: Visualization of attention from Transformer (top), TSDNet-L (middle) and TSDNet (bottom) while decoding: (a) the secend square token, and (b) the "2" in the denominator, in the corresponding mathematical expressions.

7 CONCLUSION

In this paper, we formulate image-to-markup generation as an image-to-tree generation problem, and propose TSDNet, a novel **T**ree-based **S**tructure-aware transformer **D**ecoder **Net**work to solve the problem. Different from the existing encoder-decoder based approaches that cannot effectively capture the rich structural information embedded in the target markup, TSDNet learns to generate the target markup as tree in the DFS order, and relies on a novel tree-structured attention to operate on tree structure to fully exploit the structural information. TSDNet outperforms the state-of-the-art methods by up to 5.6% in mathematical expression recognition and up to 35.34% in the chemical formula recognition. Our experimental results also prove the effectiveness of TSDNet and TSA to learn the complicated correlation from the structural information in the target markup.

ACKNOWLEDGMENTS

This work was supported, in part, by Hong Kong General Research Fund (under grant number 16200120) and Innovation and Technology Fund for Better Living (under grant number ITB/FBL/B051/20/P). We would like to thank the Turing AI Computing Cloud (TACC) [43] and HKUST iSING Lab for providing us computation resources on their platform.

REFERENCES

[1] Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. 2016. An integrated grammar-based approach for mathematical expression recognition.

Pattern Recognition 51 (2016), 135–147.

[2] Robert H. Anderson. 1967. Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics. In *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium* (Washington, D.C.). Association for Computing Machinery, New York, NY, USA, 436–459.

[3] Abdelwaheb Belaid and Jean-Paul Haton. 1984. A syntactic approach for handwritten mathematical formula recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1 (1984), 105–111.

[4] Tony Beltramelli. 2018. Pix2code: Generating Code from a Graphical User Interface Screenshot. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Paris, France) *(EICS '18)*. Association for Computing Machinery, New York, NY, USA.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. In *Computer Vision – ECCV 2020*. Springer International Publishing, Cham, 213–229.

[6] Chunyang Chen, Ting Su, Guozhu Meng, Zhenchang Xing, and Yang Liu. 2018. From UI Design Image to GUI Skeleton: A Neural Machine Translator to Bootstrap Mobile GUI Implementation. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. Association for Computing Machinery, New York, NY, USA, 665–676.

[7] Xinyun Chen, Chang Liu, and Dawn Song. 2018. Tree-to-tree Neural Networks for Program Translation. In *Advances in Neural Information Processing Systems*, Vol. 31. Curran Associates, Inc., Montréal,Canada.

[8] Leshem Choshen and Omri Abend. 2021. Transition based Graph Decoder for Neural Machine Translation. *CoRR* abs/2101.12640 (2021). arXiv:2101.12640

[9] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. 2017. Image-to-Markup Generation with Coarse-to-Fine Attention. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, Sydney, Australia, 980–989.

[10] Vincent J. Hellendoorn, Charles Sutton, Rishabh Singh, Petros Maniatis, and David Bieber. 2020. Global Relational Models of Source Code. In *International Conference on Learning Representations*. OpenReview.net, Addis Ababa, Ethiopia.

[11] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. 2017. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 2261–2269.

[12] Frank Julca-Aguilar, Harold Mouchère, Christian Viard-Gaudin, and Nina ST Hirata. 2020. A general framework for the recognition of online handwritten graphics. *International Journal on Document Analysis and Recognition (IJDAR)* 23, 2 (2020), 143–160.

[13] A. Karpathy and L. Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 3128–3137.

[14] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar Variational Autoencoder. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, Sydney, NSW, Australia, 1945–1954.

[15] Leslie Lamport. 1985. LaTeX - A document preparation system. https://www.latex-project.org/

[16] Greg Landrum et al. 2013. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling.

[17] S. Lavirotte. 1997. Optical Formula Recognition. In *2013 12th International Conference on Document Analysis and Recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 357.

[18] Minli Li, Peilin Zhao, Yifan Zhang, Shuaicheng Niu, Qingyao Wu, and Mingkui Tan. 2021. Structure-Aware Mathematical Expression Recognition with Sequence-Level Modeling. In *Proceedings of the 29th ACM International Conference on Multimedia*. Association for Computing Machinery, New York, NY, USA, 5038–5046.

[19] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. 2021. LocalViT: Bringing Locality to Vision Transformers. *CoRR* abs/2104.05707 (2021). arXiv:2104.05707

[20] Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *5th International Conference on Learning Representations*. OpenReview.net, Toulon, France.

[21] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations*. OpenReview.net, New Orleans, LA, USA.

[22] Z. Ma, C. Yuan, Y. Cheng, and X. Zhu. 2019. Image-to-Tree: A Tree-Structured Decoder for Image Captioning. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE Computer Society, Los Alamitos, CA, USA, 1294–1299.

[23] Scott MacLean and George Labahn. 2013. A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets. *International Journal on Document Analysis and Recognition (IJDAR)* 16, 2 (2013), 139–163.

[24] Mahshad Mahdavi, Richard Zanibbi, Harold Mouchere, Christian Viard-Gaudin, and Utpal Garain. 2019. ICDAR 2019 CROHME + TFD: Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE Computer Society, Los Alamitos, CA, USA, 1533–1538.

[25] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, and U. Garain. 2014. ICFHR 2014 Competition on Recognition of On-Line Handwritten Mathematical Expressions (CROHME 2014). In *2014 14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE Computer Society, Los Alamitos, CA, USA, 791–796.

[26] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, and U. Garain. 2016. ICFHR2016 CROHME: Competition on Recognition of Online Handwritten Mathematical Expressions. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE Computer Society, Los Alamitos, CA, USA, 607–612.

[27] Xuan-Phi Nguyen, Shafiq Joty, Steven Hoi, and Richard Socher. 2020. Tree-Structured Attention with Hierarchical Accumulation. In *International Conference on Learning Representations*. OpenReview.net, Addis Ababa, Ethiopia.

[28] Kohulan Rajan, Henning Otto Brinkhaus, Achim Zielesny, and Christoph Steinbeck. 2020. A review of optical chemical structure recognition tools. *Journal of Cheminformatics* 12, 1 (2020), 1–13.

[29] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 1073–1083.

[30] Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered Neurons: Integrating Tree Structures into Recurrent Neural Networks. In *International Conference on Learning Representations*. OpenReview.net, New Orleans, LA, USA.

[31] Vighnesh Shiv and Chris Quirk. 2019. Novel positional encodings to enable tree-based transformers. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc., Vancouver, Canada.

[32] Zeyu Sun, Qihao Zhu, Yingfei Xiong, Yican Sun, Lili Mou, and Lu Zhang. 2020. TreeGen: A Tree-Based Transformer Architecture for Code Generation. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 05 (Apr. 2020), 8984–8991.

[33] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 1556–1566.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., Long Beach, CA, USA, 5998–6008.

[35] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, Vancouver, BC, Canada.

[36] Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. 2018. A Tree-based Decoder for Neural Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 4772–4777.

[37] Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree Transformer: Integrating Tree Structures into Self-Attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 1061–1070.

[38] David Weininger. 1988. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences* 28, 1 (1988), 31–36.

[39] Jin-Wen Wu, Fei Yin, Yan-Ming Zhang, Xu-Yao Zhang, and Cheng-Lin Liu. 2019. Image-to-Markup Generation via Paired Adversarial Learning. In *Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, Cham, 18–34.

[40] Jin-Wen Wu, Fei Yin, Yan-Ming Zhang, Xu-Yao Zhang, and Cheng-Lin Liu. 2020. Handwritten mathematical expression recognition via paired adversarial learning. *International Journal of Computer Vision* 128, 10 (2020), 2386–2401.

[41] Binbin Xie, Jinsong Su, Yubin Ge, Xiang Li, Jianwei Cui, Junfeng Yao, and Bin Wang. 2021. Improving Tree-Structured Decoder Training for Code Generation via Mutual Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 16 (May 2021), 14121–14128.

[42] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*. PMLR, Lille, France, 2048–2057.

[43] Kaiqiang Xu, Xinchen Wan, Hao Wang, Zhenghang Ren, Xudong Liao, Decang Sun, Chaoliang Zeng, and Kai Chen. 2021. TACC: A Full-stack Cloud Computing Infrastructure for Machine Learning Tasks.

[44] Yilin Yang, Longyue Wang, Shuming Shi, Prasad Tadepalli, Stefan Lee, and Zhaopeng Tu. 2020. On the Sub-layer Functionalities of Transformer Decoder. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 4799–4811.

[45] Pengcheng Yin and Graham Neubig. 2018. TRANX: A Transition-based Neural Abstract Syntax Parser for Semantic Parsing and Code Generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Brussels, Belgium, 7–12.

[46] Jianshu Zhang, Jun Du, Yongxin Yang, Yi-Zhe Song, Si Wei, and Lirong Dai. 2020. A Tree-Structured Decoder for Image-to-Markup Generation. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, Virtual Event, 11076–11085.

[47] Jianshu Zhang, Jun Du, Shiliang Zhang, Dan Liu, Yulong Hu, Jinshui Hu, Si Wei, and Lirong Dai. 2017. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition* 71 (2017), 196–206.

[48] Daniel Zügner, Tobias Kirschstein, Michele Catasta, Jure Leskovec, and Stephan Günnemann. 2021. Language-Agnostic Representation Learning of Source Code from Structure and Context. In *International Conference on Learning Representations*. OpenReview.net, Austria.