applied sciences

MDPI

Article

Automated Recognition of Chemical Molecule Images Based on an Improved TNT Model

Yanchi Li [1], Guanyu Chen [2] and Xiang Li [1,*]

[1] School of Computer Science, China University of Geosciences, Wuhan 430079, China; liyanchi@cug.edu.cn
[2] Informatization Office, China University of Geosciences, Wuhan 430079, China; gychen@cug.edu.cn
* Correspondence: lixiang@cug.edu.cn; Tel.: +86-139-9561-9111

Abstract: The automated recognition of optical chemical structures, with the help of machine learning, could speed up research and development efforts. However, historical sources often have some level of image corruption, which reduces the performance to near zero. To solve this downside, we need a dependable algorithmic program to help chemists to further expand their research. This paper reports the results of research conducted for the Bristol-Myers Squibb-Molecular Translation competition, which was held on Kaggle and which invited participants to convert old chemical images to their underlying chemical structures, annotated as InChI text; we define this work as molecular translation. We proposed a model based on a transformer, which can be utilized in molecular translation. To better capture the details of the chemical structure, the image features we want to extract need to be accurate at the pixel level. TNT is one of the existing transformer models that can meet this requirement. This model was originally used for image classification, and is essentially a transformer-encoder, which cannot be utilized for generation tasks. On the other hand, we believe that TNT cannot integrate the local information of images well, so we improve the core module of TNT—TNT block—and propose a novel module—Deep TNT block—by stacking the module to form an encoder structure, and then use the vanilla transformer-decoder as a decoder, forming a chemical formula generation model based on the encoder–decoder structure. Since molecular translation is an image-captioning task, we named it the Image Captioning Model based on Deep TNT (ICMDT). A comparison with different models shows that our model has benefits in each convergence speed and final description accuracy. We have designed a complete process in the model inference and fusion phase to further enhance the final results.

Keywords: vision transformer; image captioning; TNT; automatic text annotation of chemical images; Bristol-Myers Squibb-Molecular Translation

Citation: Li, Y.; Chen, G.; Li, X. Automated Recognition of Chemical Molecule Images Based on an Improved TNT Model. *Appl. Sci.* **2022**, *12*, 680. https://doi.org/10.3390/app12020680

Academic Editor: Antonio Fernández

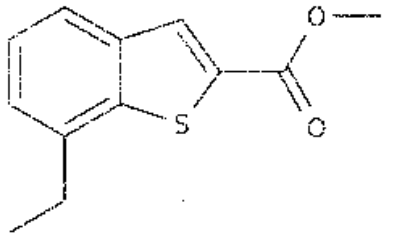
Received: 2 November 2021
Accepted: 6 January 2022
Published: 11 January 2022

1. Introduction

The Bristol-Myers Squibb Molecular Translation competition was designed to help chemists expand access to collective chemical research, speeding up research and development efforts in key fields by avoiding repetition of previously published chemistries and identifying novel trends via mining large data sets [1]. In this competition, participants are asked to interpret old chemical images. The objective is to convert an outsized quantity of synthetic image information into the underlying chemical structure and annotated it as International Chemical Identifier (InChI) text, which is a string jointly developed by the International Union of Pure and Applied Chemistry (IUPAC) and the National Institute of Standards and Technology (NIST) to uniquely identify the IUPAC name of a compound. The string comprises four parts: (1) version, (2) chemical formula, (3) atomic linkage layer and (4) hydrogen layer. A chemical image and its corresponding InChI text in the training set are shown in Figure 1.

In this work, we use a deep learning methodology to accomplish the molecular translation. The general process is to input chemical images into the model. Then the model

generates the corresponding annotations. The molecular translation is essentially an image-captioning task, which is a cross between the fields of computer vision and natural language processing (NLP). In computer vision, convolutional architectures remain dominant [2–5]. To take advantage of the powerful convolutional structure, we designed a model for use as a comparison, which consists of a convolutional neural network (CNN) [4,5] as an encoder and a recurrent neural network (RNN) [6] as a decoder (see Section 4.1.2 for details).



(1) Version

(2) Chemical Formula

(3) Atomic Layer

(4) Hydrogen Layer

InChI=1S/C12H12O2S/c1-3-8-5-4-6-9-7-10(12(13)14-2)15-11(8)9/h4-7H,3H2,1-2H3

(1) (2) (3) (4)

Figure 1. A glimpse of the training set.

NLP is the bridge between machine language and human language for the purpose of human–machine communication. Today, the most fundamental model for handling NLP tasks is the transformer, which is a network mainly composed of Multi-head Self-Attention (MSA) [7], which has beaten the prevailing best models based on recurrent and convolutional structures on machine translation tasks using an attention mechanism. The training time was significantly reduced as the encoder side can calculate in parallel.

Nowadays, transformers are widely used in the NLP sector [8,9]. Researchers have also applied this efficient structure to the field of computer vision, Vision transformer (ViT) [10] accomplishes vision tasks without much modification of vanilla transformer. ViT divides an image into a series of patches. On the basis of ViT, models based entirely on self-attention have emerged in recent years [11–13]. TNT [12] follows the way of image processing in ViT and its authors point out that ViT cannot model the local information of an image, and a new structure is proposed: TNT block, which solves the above problems. However, we found that TNT could not finely integrate the local information in chemical images, so we propose the Deep TNT block. We stack Deep TNT blocks and use the native transformer-decoder for decoding to obtain a novel generative model, which we call ICMDT (the model structure is shown in Figure 2). The final result is evaluated by the Levenshtein distance, which is a metric used to measure the similarity of two sequences. This metric primarily calculates the minimum number of single-character edit operations needed for interconversion between two sequences.

To demonstrate the effectiveness of ICMDT, in our work, we analyze three aspects of the model size, the convergence speed during training, the final metrics (the mean Levenshtein distance between inference results and the ground truth InChi values, the entire evaluation process is conducted on the Kaggle platform). The influence of image pre-processing and post-processing on the final results is also analyzed.

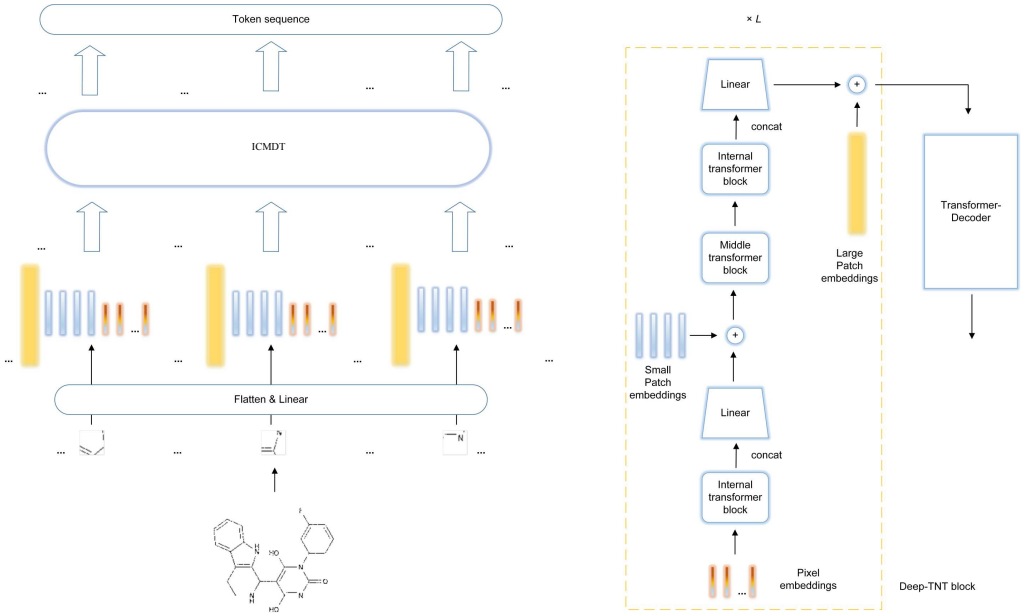


Figure 2. Simplified processing flow of the molecular translation task and the structure of ICMDT.

2. Related Work

2.1. Image Captioning

Image captioning is a typical application in the field of multimodal data processing [14]. Traditional image-captioning algorithms are divided into two categories: (1) template-based image-captioning algorithms [15–17] and (2) retrieval-based image-captioning algorithms [18–20]. The former limits the diversity of description texts to some extent and does not allow a generation of variable-length description statements. The description statements generated by the latter method are more flexible, and the method constructs a data retrieval database rich in semantic information by collecting a large number of images and their corresponding text descriptions, but it is limited by the existing retrieval database.

Image-captioning algorithms based on deep learning architecture can be broadly classified into two categories. The first includes basic encoding-decoding image-captioning algorithms [21–23]. In this framework, the encoding process extracts the visual features of the target image by using a deep convolutional neural network (DCNN), and the decoding process translates the obtained visual features into description statements by RNN. However, the descriptive statements generated by these models cannot be linked to the relevant regions of the image. The second category includes the attention-based encoding-decoding image-captioning algorithm [24–27]. The algorithm introduces an attention mechanism in the decoding stage that combines the words generated in the previous moment with the local visual information of the image, making it possible to dynamically focus on different regions of the image as each word is generated by the language module. In recent years, some work has also been carried out on the basis of the self-attention model [9] to compensate for the shortcomings of the transformer model for image-captioning tasks and to further improve the model's performance [28–30].

2.2. Transformer

The transformer structure (Vaswani et al., 2017) has become the standard paradigm in NLP, and a series of excellent models have been derived from it. BERT [8] is the most representative. It is pre-trained by denoising the text corpus with self-supervision and is then placed on a downstream task for fine-tuning. The transformer's powerful feature learning capability and bi-directional encoding via masked language models dramatically improves the benchmark performance of various NLP tasks.

Given its powerful learning capabilities, BERT has been applied to cross-modal domains since 2019, aiming to build a generic feature learning model for cross-modal domains. Several experiments have shown that architectures such as BERT continues to have strong learning capabilities in cross-modal domains. Its application in cross-modal domains has been divided into two schools of thought: one is the single-stream model [31–35], in which language information and vision information are amalgamated at the start, then fed directly into the encoder (transformer) together. The other is the dual-stream model [36,37], in which language information and vision information are first passed through two separate transformer-encoder modules. Then the cross-transformer module achieves the fusion of different modal information. VLP [35] introduces a decoder and shares parameters with an encoder, it is the first type of model to be used for generation tasks (Image captioning).

In computer vision, the transformer model has inspired improvements in the CNN architecture [38–40]. For example, self-attention has been integrated into the CNN architecture [41–44], and in recent years, increasing transformer architectures have been put into rich computer vision tasks (target detection, semantic segmentation) [45–48], ViT [10] achieved the image classification task by a standard transformer-encoder with few modifications, and a series of vision transformers followed under the influence of that work [11–13,45,48,49]. DeiT [11] introduces distillation and some training strategies based on ViT [10] to improve training speed and accuracy.

TNT [12] processes each image patch through two transformer blocks (inner transformer block, outer transformer block). The pixel-level information is fused with the patch-level information. The model outperforms DeiT on ImageNet. In this paper, the TNT block (the core module of TNT) is further improved so that it can both capture local information in the image and integrate local information in a more precise way. It achieves the best trade-off in size, speed, and performance compared to four other baseline models.

3. Materials and Methods

In this section, we introduce the preliminaries of transformers and vision transformers, as well as our improvements based on the TNT block and our network structure, position encoding.

3.1. Transformer

The transformer encoder consists of MSA, Feed-Forward Networks (FFN), Layer Normalization (LN), and skip connections. In general terms, a transformer consists of two parts: an encoder and a decoder.

In the base module of MSA, scaled dot-product attention, model inputs and the three learnable parameter matrices are computed in dot product form and yield three vectors of three types: queries (Q), keys (K) and values (V) (Q has the same dimension as K). The three matrices are $W_Q \in \mathbb{R}^{d \times d_k}$; $W_K \in \mathbb{R}^{d \times d_k}$; $W_V \in \mathbb{R}^{d \times d_v}$ where d, d_k and d_v are the dimensions of input and Q, K, V. The subsequent calculations revolve around Q, K, V (as shown in Equation (1)).

$$\text{Attention}(Q, K, V) = \text{sotfmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

The calculation in Equation (1) yields an attention result, while MSA is responsible for computing multiple attentions in parallel: MSA divides the attention into h parts, performs the operations in parallel, merges the attention results of each part and obtains the result after a linear layer projection.

Both the MSA block and FFN are immediately followed by a LN and skip connection. The decoder differs from the encoder by the addition of a masked self-attention and cross-attention mechanism. In masked self-attention, attention is restricted so that the queries at each position can only focus on key-value pairs at the current and previous positions. The

last part we call the illegal position. The conventional practice is to add a mask matrix to the attention score, where illegal positions are masked using the following formula:

$$A_{ij} = -\infty \text{ if } i < j^2 \tag{2}$$

Cross-attention differs from self-attention in that queries are obtained from the output projection of the previous (decoder) layer, while keys and values are obtained using the output projection of the encoder.

3.2. Vision Transformer

As the pioneer of vision transformer, ViT [10] differs from the vanilla transformer-encoder in that the FFN is replaced with a multilayer perceptron (MLP) and the LN is placed in front of the MSA block and the MLP block. The only difference between the MLP and the feed-forward layer is that the activation function is changed from ReLu to GeLu [50]. In image processing, ViT segments a 2D image into a series of patches $\mathcal{X} = [X_1, X_2, X_3, \ldots, X_n] \in \mathbb{R}^{n \times p^2 \times c}$, where n represents the length of the sequence of facets; p^2 represents the size of each patch, and c represents channels. These patches are then flattened and projected through a linear layer into a 1D sequence $\mathcal{X}_f \in \mathbb{R}^{n \times d}$. After adding a class token and attaching a position encoding $E_{\text{pos}} \in \mathbb{R}^{(n+1) \times d}$, they are fed into the model for processing (Equations (3) and (4)), and the classification results are output through an MLP.

$$z^*_{\ell-1} = MSA(LN(z_{\ell-1})) + z_{\ell-1} \quad \ell = 1, 2 \ldots L \tag{3}$$

$$z_\ell = MLP\left(LN\left(z^*_{\ell-1}\right)\right) + z^*_{\ell-1} \quad \ell = 1, 2 \ldots L \tag{4}$$

3.3. Deep-TNT Block

TNT block [12] models local information and global information separately. First, the local information (pixel embedding) is processed by the inner-transformer block, and then the local information is attached to the corresponding patches and the outer transformer block to process the global information (patch embedding). Path-level information is integrated into global information through simple position encoding, but we believe this operation is not an accurate way to obtain global information. Thus, to better model the global and local information of chemical images, we model the patch-level information again. First, we fuse several patch-level messages into one large patch, then process them through a transformer-block. This makes the integration of location coding and global information much easier. We denote this module as Deep TNT block. It is used to process three levels of information (as shown in Figure 3), and by stacking a total of L times, this module forms the encoder of our model.

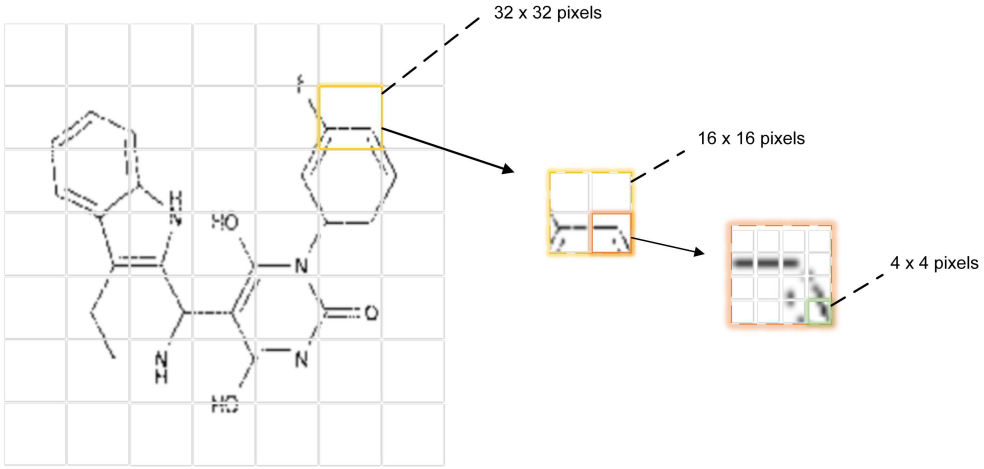


Figure 3. Processing of three levels of information.

Similarly, we divide an image into n non-overlapping patches, noting that the size of the patches (p, p) here is larger than that in TNT. Then we unfold and linearly project each patch into a sequence of patch tensors of size (p', p'); Z_0^i denotes the patch tensor:

$$\mathcal{Z}_0 = \left[Z_0^1, Z_0^2, \ldots, Z_0^n\right] \in \mathbb{R}^{n \times p' \times p' \times c} \tag{5}$$

where Z_0^i acts as a sequence of small patch embeddings; c represents the channel. Then each small patch is further divided into target sizes (p'', p'') and linearly projected again into a sequence of pixel-level tensors, where the sequence length is kept as n, which is for easier fusion with higher-level information.

$$y_0 = \left[Y_0^1, Y_0^2, \ldots, Y_0^n\right] \in \mathbb{R}^{n \times p'' \times p'' \times c_1} \tag{6}$$

In this case, y_0 represents the whole picture, and Y_0^i is a small patch level of information consisting of pixel-level information. After re-division, to keep the length of the sequence, the number of channels is changed. This is represented by c_1, in the same way as in TNT. Here we consider each tensor Y_0^i as a pixel embedding sequence:

$$Y_0^i = \left[\mathcal{Y}_0^{i,1}, \mathcal{Y}_0^{i,2}, \ldots, \mathcal{Y}_0^{i,m}\right] \tag{7}$$

where $m = p''^2$; $\mathcal{Y}_0^{i,j} \in \mathbb{R}^{c_1}$, and $j = 1, 2, \ldots, m$.

We process different levels of information by three data streams. The first one is used to process large patch embeddings; the second one is used to process small patch embeddings, and the third one processes pixel-level features in small patches. The transformer blocks in the three processes are the exterior transformer block, middle transformer block, and internal transformer block.

$$Y_\ell^{*i} = Y_{\ell-1}^{*i} + MSA\left(LN\left(Y_{\ell-1}^{*i}\right)\right) \tag{8}$$

$$Y_\ell^i = Y_\ell^{*i} + MLP\left(LN\left(Y_\ell^{*i}\right)\right) \tag{9}$$

Equations (8) and (9) show the processing of the pixel embedding sequence in the internal transformer block, which is consistent with ViT in that the LN is also preceded. Here, subtle local information is modeled, and each vertex in the chemical structure and its connected chemical elements are captured in one place where ℓ represents the ordinal number of the ℓth level; $\ell = 1, 2, \ldots, L$, and we denote the total number of layers as L.

$$Z_{\ell-1}^i = Z_{\ell-1}^i + Flat\left(Y_{\ell-1}^i\right)\mathcal{W}_{\ell-1} + b_{\ell-1} \tag{10}$$

$$Z_\ell^{*i} = Z_{\ell-1}^i + MSA\left(LN\left(Z_{\ell-1}^i\right)\right) \tag{11}$$

$$Z_\ell^i = Z_\ell^{*i} + MLP\left(LN\left(Z_\ell^{*i}\right)\right) \tag{12}$$

Equations (10)–(12) show the process of integrating pixel-level features into the small patch embedding and modeling the small patch embedding sequence. The feature information output by the internal transformer block is expanded into a vector by the $Flat()$ function and then processed by a linear layer to be attached to the corresponding small patch embedding. $\mathcal{W}_{\ell-1} \in \mathbb{R}^{m \times c_1 \times \left(p'^2 \times c\right)}$ and $\mathbf{b}_{\ell-1} \in \mathbb{R}^{p'^2 \times c}$ are the weights and bias, respectively. The sequence of small patch embeddings is modeled with an exterior transformer block.

We use $\mathcal{N}_0 = \left[N_0^1,\ N_0^2, \ldots, N_0^n\right] \in \mathbb{R}^{n \times d}$ to denote the sequence of large patch embeddings. Since it is no longer used as a classification task and is simply encoded to obtain the

intermediate vectors, classification markers are eliminated here. As in Equations (10)–(12), the small patch tensor is first expanded and added to the patch embedding by linear projection.

$$\mathcal{N}_{\ell-1}^{i} = \mathcal{N}_{\ell-1}^{i} + F\,\mathrm{lat}\left(Z_{\ell}^{i}\right)\mathcal{W}_{\ell-1}' + b_{\ell-1}' \tag{13}$$

$$\mathcal{N}_{\ell}^{*i} = \mathcal{N}_{\ell-1}^{i} + MSA\left(LN\left(\mathcal{N}_{\ell-1}^{i}\right)\right) \tag{14}$$

$$\mathcal{N}_{\ell}^{i} = \mathcal{N}_{\ell}^{*i} + MLP\left(LN\left(\mathcal{N}_{\ell}^{*i}\right)\right) \tag{15}$$

Here, $N_{\ell-1}^{i} \in \mathbb{R}^{d}$; $\mathcal{W}_{\ell-1}' \in \mathbb{R}\left(p'^{2} \times c\right) \times d$; $\mathbf{b}_{\ell-1}' \in \mathbb{R}^{d}$. The sequence of patch embeddings is modeled using an exterior transformer block.

We built the Deep-TNT as the encoder part of our generative model by fusing the local information at a finer level. The internal transformer block makes pixel-level interactions to extract deep local image information. The middle transformer block extracts "shallow global features" from a sequence of small patches, and the exterior transformer block further fuses the shallow global features to obtain more accurate "deep global features". For the whole Deep-TNT block, the outputs and inputs include pixel embeddings, small patch embeddings, and large patch embeddings, so it can be expressed in general as

$$y_{\ell}, \mathcal{Z}_{\ell}, \mathcal{N}_{\ell} = Deep - TNT(y_{\ell-1}, \mathcal{Z}_{\ell-1}, \mathcal{N}_{\ell-1}) \tag{16}$$

3.4. Position Encoding

To better integrate patch-level features and pixel-level features to obtain complete information about the image, position encoding is essential. Positional information is incorporated in the form of fixed or trainable [51] position embeddings.

They are added to the patch tokens before the first transformer block and are then fed into a stack of transformer blocks. The details are shown in Figure 4. We add corresponding position encoding to each small patch embedding and pixel embedding, which are shared and kept consistent between each large patch. The difference is that for the large patch embedding, a learnable 1D position encoding is used here.

$$\mathcal{N}_{0} \leftarrow \mathcal{N}_{0} + E_{\text{large patch}} \tag{17}$$

$$Z_{0}^{i} \leftarrow Z_{0}^{i} + E_{\text{small patch}} \tag{18}$$

$$Y_{0}^{i} \leftarrow Y_{0}^{i} + E_{\text{pixel}} \tag{19}$$

where $E_{\text{large patch}} \in \mathbb{R}^{n \times d}$ is the position encoding for large patches; $E_{\text{small patch}} \in \mathbb{R}^{p'^{2} \times c}$ is the position encoding for small patches, and $E_{\text{pixel}} \in \mathbb{R}^{m \times c_1}$ is the fixed pixel position encoding.

3.5. Parameter Setting in the Network Architecture

The middle transformer block and the internal transformer block in our Deep TNT are similar to the structure of the TNT block. We considered the lightness of the model, so we set the hyperparameters of our two transformer blocks according to the configuration of TNT-S. To maximize the utility of TNT-S, we kept its structural integrity and modified only the embedding dimension. The small patch size was set to 16 × 16 and the pixel-level patch size p'' was set to 4.

In Table 1, "Hidden_size" and "Mlp_act" represent the dimension of the hidden layer in the MLP and the activation function used in the MLP, respectively. "Vocab size" refers to the maximum length of the word list generated by the decoder, and "text_dim" represents the embedding dimension of the label into the decoder.

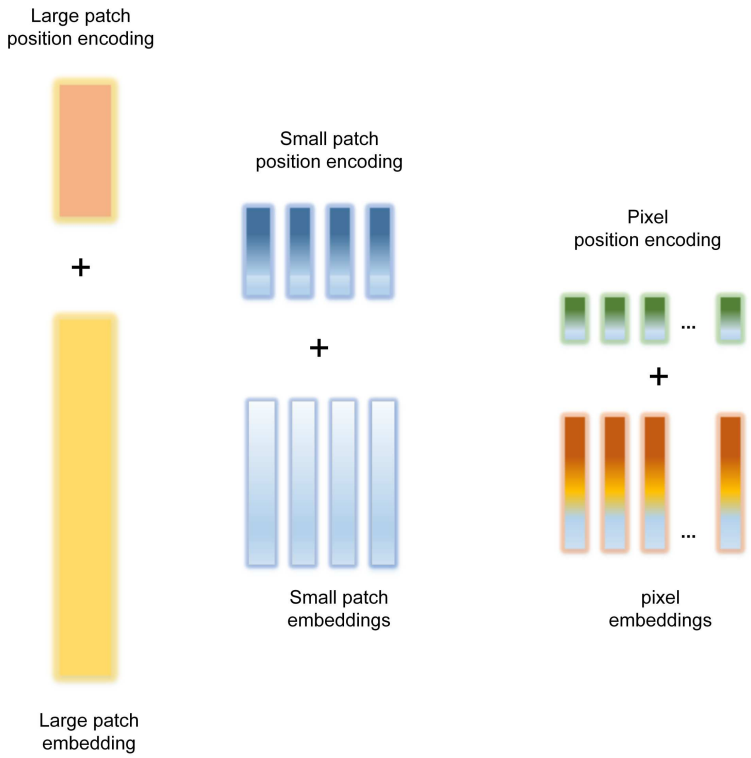


Figure 4. Patch embedding and Pixel embedding position encodings.

Table 1. Strategies used in the training phase.

Encoder						
	Dim #	Heads	Pixel/batch size	Depth	Hidden_size	Mlp_act
Internal transformer block	160	4	4	12	640	GELU
Middle transformer block	10	6	16		128	GELU
Exterior transformer block	2560	10	32		5120	GELU
Decoder						
	Decoder dim	Heads	FFN dim	Depth	Vocab_size	text_dim
Transformer-decoder	2560	8	1024	3	193	384

4. Results

In this section, we first discuss the experimental setup, such as dataset, optimization, pre-processing, and training methods, and then compare five models from multiple perspectives to show the efficiency of ICMDT. Finally, we conclude with a complex and complete model inference and fusion process.

4.1. Setup

4.1.1. Datasets

The dataset used in this experiment is the bms-molecular-translation provided by Bristol-Myers Squibb. The dataset is divided into a training set and a test set. The training set contains 2,424,186 chemical images. Each image has InChI markers of different lengths according to the complexity of the chemical structure. Thus, training the model on this dataset is a supervised learning process. To simulate the real old chemical images, image noise was added to the images, and some images were blurred, which added difficulty to the training. The test data contain 1,616,107 chemical images. Compared with the training set, some of the images in the test set are noisier and have a much different aspect ratio.

4.1.2. Models for Comparison

We constructed several representative models for comparison. In the first type of convnet is the encoder and the full transformer structure is the decoder because it maintains the integrity of the ViT structure. ViT usually segments the image into a series of patches and then forms the image patch embeddings through a linear layer, which serves as the input to the model. We instead obtained the model input through the convolution function, where the convnet and transformer-encoder can be analogous to ViT. Thus, the whole model can be seen as a combination of ViT and the transformer-decoder (as shown in Figure 5). Another type of model is the combination of convnet and RNN. We added the attention mechanism in the decoder part of the model (as shown in Figure 6).

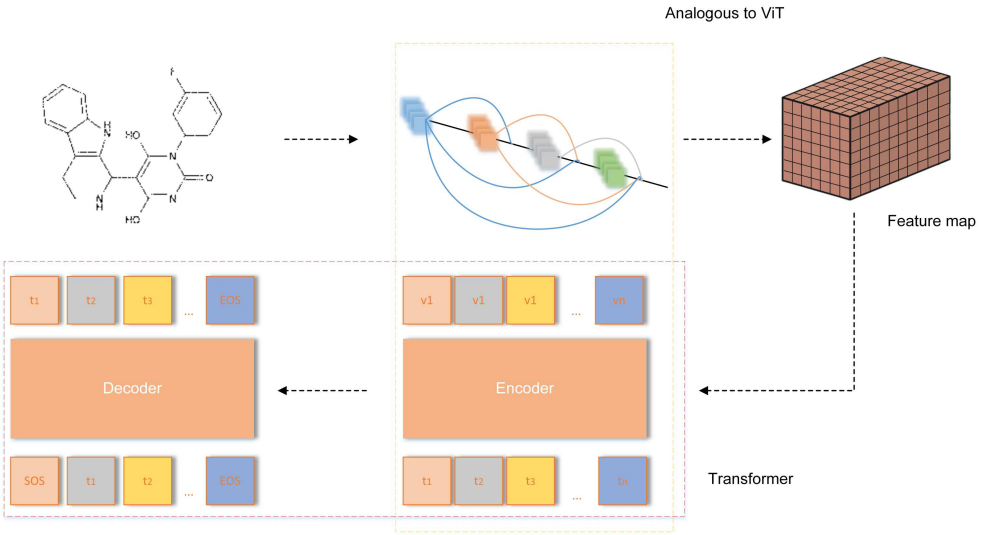


Figure 5. The combination of convnets and transformer. The red box represents the complete transformer structure, and the yellow box is the structure we use as an analogy to ViT. Note that here we use ResNet to symbolize convnets; in the actual experiment, we had two options here, ResNet and EfficientNet (the same in Figure 6).

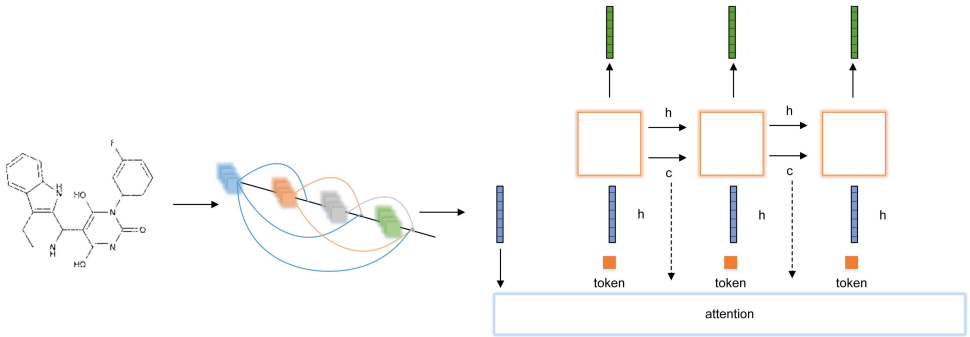


Figure 6. The structure of the model combined by convnet and RNN. The orange box represents the RNN structure. The 3D chemical image is processed by convnet to generate the feature vector and is processed by the attention mechanism to generate the attention vector for the token sequence, and the attention vector is fused with the token embedding. The state vector h, generated by the RNN at the previous time step, is also processed by the attention mechanism and fused with the previous two. It is used as the input for the current time step.

4.1.3. Pre-Processing

In terms of pre-processing, both the training and test sets have too much noise, and the core chemical structure is only in the middle of the image (blank borders exist in the image).

We adopted three means to process the image: (1) image denoising (noise filtering in images employing convnet at a convolution kernel size of 1), (2) smart cropping (the blank

border with little information about the chemical structure is removed), and (3) padding resizing (during the training process, resizing images with inconsistent aspect ratios may cause image distortion and loss of original information). We first reshaped the image into a square, and then used the longer edge of the original image as the edge length of the new image, and the insufficient part was supplemented by the pixel points in the middle part of the image. In Section 4.3, we experiment with different combinations of these processing methods. The effects of the three pre-processing means are shown in Figure 7.

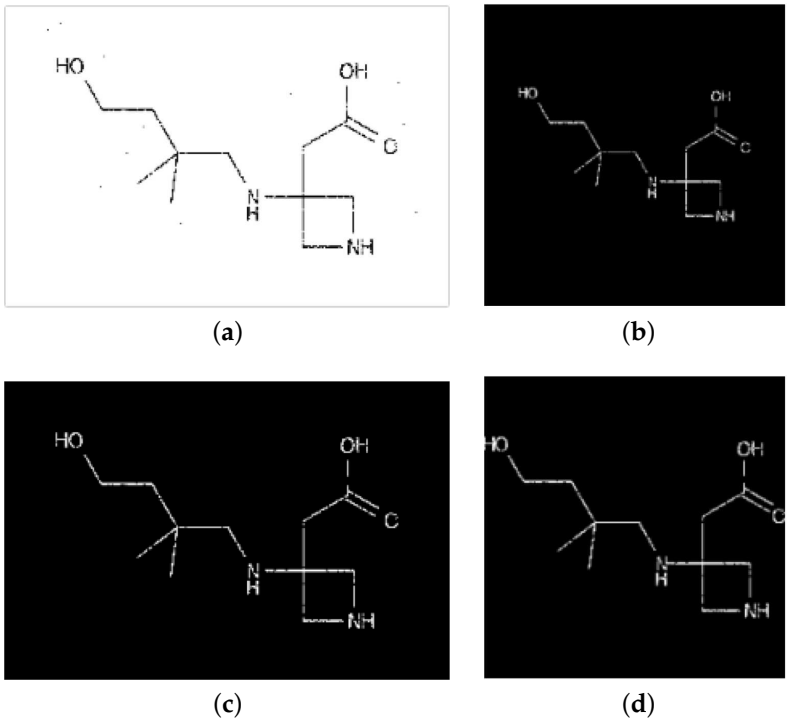


Figure 7. Comparison of original image to the image after pre-processing. (**a**) Original image. (**b**) Denoising and padding reshaping. (**c**) After denoising. (**d**) Denoising and padding reshaping, smart cropping.

4.1.4. Training Details

The experiments were all performed with ICMDT as a benchmark.

The general data augmentation methods we used without additional statements were GaussNoise, Blur, RandomRotate90 [52], and PepperNoise. To obtain the best possible model performance in a limited time, the optimizers we used were Lookahead [53] and RAdam [54]. In terms of loss function, we chose Anti-Focal loss [55], which is based on Focal loss [56]. Focal loss solves the situation of uneven distribution of samples with different training difficulties and puts more attention on error-prone (difficult to train) samples during training through weight assignment. Raunak et al. [55] pointed out that in the Seq2Seq, there is a difference between the inference within the training part and the testing part. Using this loss function can reduce this difference. In addition to this, we added label smoothing [57].

In the training process, we first trained all images at a resolution of 224 and then put images with label lengths greater than 150 to a resolution of 384 for fine-tuning, following the strategy of Touvron et al. [58]. While changing the resolution, if we did not change the patch size which will result in a change in sequence length—we used the same strategy as Dosovitskiy et al. [10] to add a new position code to the patch sequence by performing 2D interpolation of the pre-trained position encoding at low resolution.

Test time augmentation (TTA) was used to generate two model results by inference on the original test set, and the test set was processed by rotating 90° in any direction. The

two model results that were consistent and conformed to the InChI form (here using the open-source cheminformatics and machine learning toolkit, RDKit [59], to carry out the filtering) were used as pseudo-labels. To further enhance the robustness of the model and to allow the model to correctly predict subsequent tokens despite the incorrect prediction of a token during inference, we introduced Noisy Labels to randomly replace a chemical element of the corresponding label of the training samples with a certain probability. The main strategies adopted in the training and the hyperparameter settings are explained in Table 2.

Table 2. Strategies used in the training phase.

Pre-Processing	Optimizer	Batch Size	Learning Rate	Label Smooth	Loss Function	Data Augmentation
Image denoising	Lookahead ($\alpha = 0.5, k = 5$)	16	1e-4	Anti-Focal loss ($\gamma = 0.5$)	GaussNoise	
Smart cropping	RAdam ($\beta_1 = 0.9, \beta_2 = 0.99$)	32	2e-4	/	/	RandomRotate90
Padding resizing	/	64+	(4e-4)+	/	/	PepperNoise (SNR = 0.996)

In Table 2, we set the batch size from 16 to 64+. We set a smaller batch size at the beginning of the training, then gradually increased it as the training proceeded, increasing the learning rate in the same proportion [60]. When the validation loss value tended to level off, we kept the batch size constant and then slowly reduced the learning rate until it was close to the global optimum. Where 64+ means that continued growth in powers of 2 based on 64, with a maximum value of 1024, (4e-4)+ in the same way. In terms of data enhancement, we only included the three core methods. Adding pepper noise to the training data at Signal-to-noise ratio (SNR) = 0.996, the training data can fit the test data to the maximum extent and the best results. The pre-processing methods shown in Table 2 are not used by default.

4.2. Comparison

In this section, we compare the five models in terms of three dimensions: model parameters, the speed of convergence, and the best results. In Figure 8, we compare the parameters of the five models. Among them, RNN refers to the bidirectional LSTM structure. Transformer adheres to its feature of having a large number of parameters. We can see that the overall number of model parameters is lower when using RNN as the decoder. In the case of using transformer as the model's decoder, the number of parameters is the largest when using DCNN as the encoder, and the overall model parameter number is also larger than ICMDT when using the lightweight convnet as the encoder. Thus, from the overall situation, the number of model parameters for our ICMDT is at an intermediate level.

We compared the convergence process of the five models, as shown in Figure 9. Using lightweight convnet as an encoder converges faster than using DCNN as an encoder, provided that a bidirectional LSTM is used as the model decoder (the former inference on the test set at 4.9 epochs of training achieves an edit distance of 10 from the real result, while the latter training at 10.36 epochs), and the final results are better. When the transformer is the decoder, using lightweight convnet as the encoder converges faster than DCNN, but the final result of the latter is slightly better when the training time is long enough. Our model outperformed the best results of the other four models as early as 6.7 epochs, and finally converged at 9.76 epochs. We can see that our model outperformed the other four models in both convergence speed and final results.

Considered together through these three dimensions, our model is the most eclectic choice among the five models.

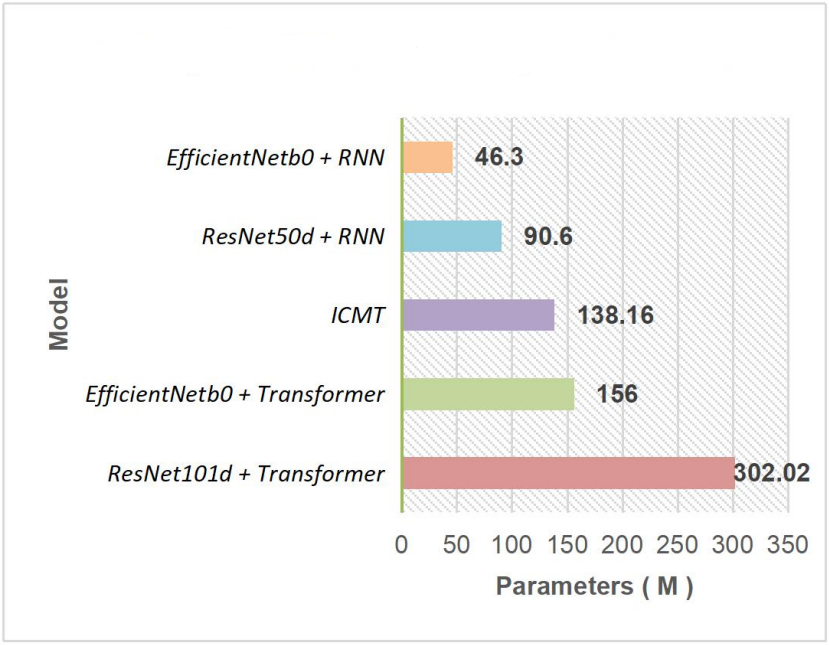


Figure 8. Comparision of model parameters.

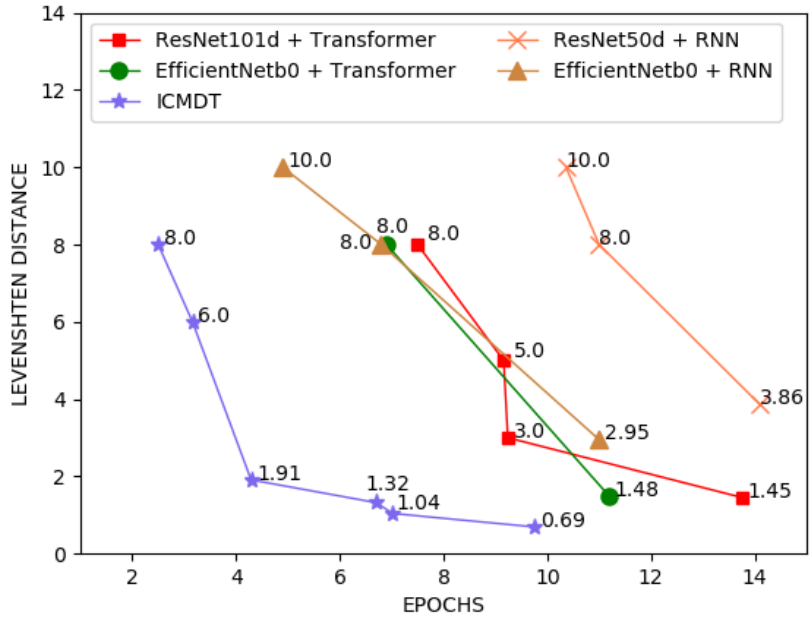


Figure 9. Convergence process comparison.

4.3. Ablation

In this section, we discuss whether fusing patch-level features is effective, whether positional encoding contributes to model performance, and the impact of three pre-processing approaches on model performance.

To explore the validity of our patch-level feature fusion and large patch position encoding, we set two comparisons: (1) change the encoder in our model to TNT-S, which we denote as TNTD; (2) keep the structure of TNTD unchanged and set the patch size to 32 to directly replace the processing conducted by the exterior transformer block, which we denote as TNTD-B. Both models use the same position encoding as TNT. We also divide ICMDT into two models. The first model is the original model, keeping our original three position encodings. In the second model we remove the position encoding used for large patches, which is denoted as ICMDT*.

From previous work [10,12] we know that adding position encoding for pixel-level sequences and small patch sequences is valid, so we do not discuss it here. From Table 3, we know that adding position encoding for large patch sequences is valid. Setting the patch size to the size of our large patch directly on the basis of TNT will reduce the performance of the model (TNTD vs. TNTD-B), and from the comparison of ICMDT vs. TNTD we conclude that further fusion of patch information can improve the model's performance. In fact, other competitors in the Bristol-Myers-Squibb competition often used a fusion of several excellent models as their final submission, but our single ICMDT outperformed them.

Table 3. A discussion of position encoding as well as feature fusion.

Model	Parameters (M)	Levenshtein Distance
ICMDT	138.16	0.69
TNTD	114.36	1.29
TNTD-B	114.36	1.37
ICMDT*	138.16	1.04

We further investigated data pre-processing (training data denoising of all data), smart cropping, padding resizing, and test data denoising (all data). Then, we conducted experiments on the fraction of denoised data within the training data for the purpose of deriving the best mixture of original data and denoised data in the training set.

From Table 4, we can see that if we only denoise all the data in the training set, the performance of the model is greatly reduced, but if we also denoise the data in the test set, the performance of the model improves. The model's performance is not affected by filling the training data with padding resizing while keeping all the training and test data denoised, so a simple resizing of the images does not lead to much information loss. However, we find that smart cropping of the images degrades the model's performance, presumably because the white edges in the images also contain valid information.

Table 4. Study of the ratio of denoised images to original training data.

Pre-Processing Strategy	Training Data Denoising	Smart Cropping	Padding Resizing	Test Data Denoising	Levenshtein Distance
Group 1	√	×	×	×	7.69
Group 2	√	×	×	√	2.69
Group 3	√	√	√	×	7.82
Group 4	√	√	√	√	3.02
Group 5	√	×	√	×	7.69
Group 6	√	×	√	√	2.69

We experimented with the proportion of denoised data in the training set and whether to denoise the test data; note that the other two pre-processing methods are not included in that experiment.

As shown in Table 5, we set the ratio of denoised data to original data in the training data to 2:13. When we also denoise the test data, we can achieve the best result of hybrid training, so we can assume that the performance of the model will not be damaged too much when denoising is used as a data augmentation method. Since there is more noise within the test data than within the training data, we find that adding noise to the data during training better preserves the best performance of the model because it can fit the test data to the maximum extent. Note that we did not do any denoising of the test and training data during the training and testing of the final model. The realistic chemical images will have more noise. Although we cannot be sure whether these noises have meaningful information, it is certain that blindly denoising an image is not the best option.

Table 5. Pre-processing strategy study.

Denoised Training Data + Original Training Data						
Proportion	13:2	11:4	8:7	7:8	4:11	2:13
Denoised test data	7.5	6.32	5.02	5.0	2.69	1.04
Original test data	2.57	1.87	2.54	2.56	4.87	6.64

4.4. Analysis of Inference Results

In this section, the differences in inference results between ICMDT and several other comparative models on chemical images of varying complexity will be presented and the strengths and weaknesses analyzed.

We selected three representative chemical images, divided into three levels of complexity: low, general and high. The inference results of the model for the three images are represented in the inference results of the model for chemical images of the same complexity.

The true InChI expressions corresponding to the three chemical structures in Figure 10 are shown in Table 6.

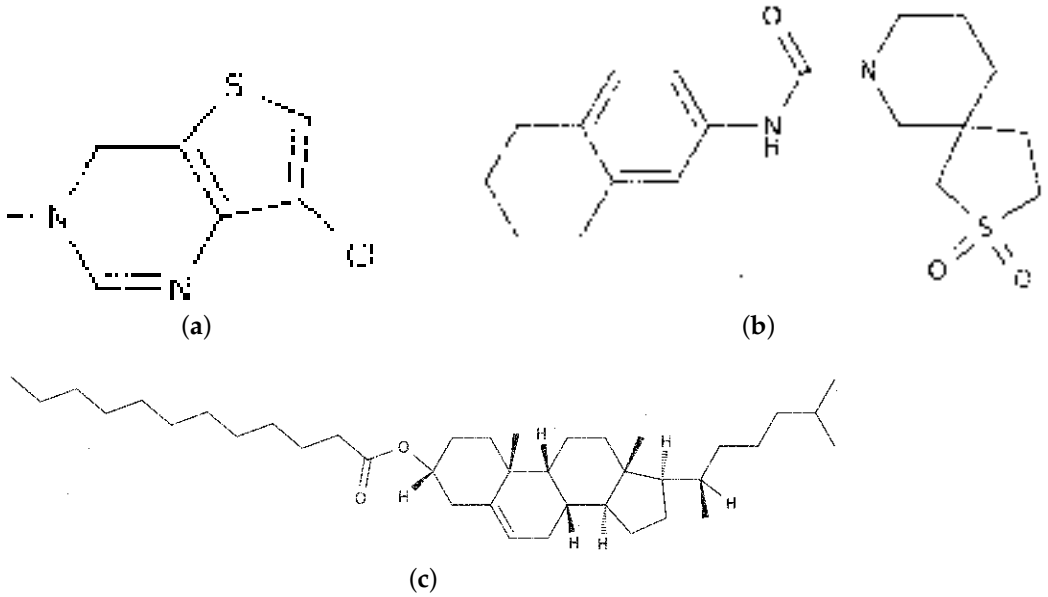


Figure 10. Example diagrams with varying levels of complexity. (**a**) Example diagram of low complexity. (**b**) Example diagram of general complexity. (**c**) Example diagram of a high level of complexity.

The information in InChI is structured as a sequence of layers and sub-layers, with each layer providing one specific type of information. The layers and sub-layers are separated by the delimiter "/" and start with a characteristic prefix letter. In view of these features, the prefixes "/C", "/C", "/h", "/t" of each layer adopt fixed encoding mode and divide these prefixes consisting of two characters into a token when dividing labels. Therefore, our model and other comparative models will not incorrectly predict these prefixes when reasoning. We just need to pay attention to the specific contents of each layer: atoms, the number of atoms, connections between atoms, etc.

In Table 6, the inference results of the six models for the chemical structures in Figure 10a are as follows: TNTD, our model and the two combined models of "ResNet101d + Transformer" and "EfficientNetb0 + Transformer" can correctly predict the InChI expression of the image. In fact, these four models are very reliable for such simple chemical structure predictions. The result of the "ResNet50d + RNN" model shows that the number of some atoms in the atomic layer is wrong, and the "EfficientNetb0 + RNN" model is wrong to predict the content in the hydrogen layer. However, in a large number of chemical

structures of the same type, the two models incorrectly predict the hydrogen and atomic layers in only a few cases, and correctly predict in most cases. The analysis of the overall situation shows that the model with a transformer as the decoder is more advantageous.

Table 6. Results of ICMDT and other comparative models for the inference of three types of chemical structures.

Image Number & Real InChI	Model	Inference Results
Figure 10a InChI = 1S/C7H7C1N2S /c8-7-9-4-6-5(10-7) 2-1-3-11-6/h4H,1-3H2	ICMDT	InChI = 1S/C7H7ClN2S/c8-7-9-4-6-5(10-7)2-1-3-11-6/h4H,1-3H2
	ResNet101d + Transformer	InChI = 1S/C7H7ClN2S/c8-7-9-4-6-5(10-7)2-1-3-11-6/h4H,1-3H2
	EfficientNetb0 + Transformer	InChI = 1S/C7H7ClN2S/c8-7-9-4-6-5(10-7)2-1-3-11-6/h4H,1-3H2
	ResNet50d + RNN	InChI = 1S/C7H7ClN2S/c8-7-9-4-6-5(10-7)2-1-9-11-8/h4H,1-3H2
	EfficientNetb0 + RNN	InChI = 1S/C7H7ClN2S/c8-7-9-4-6-5(10-7)2-1-3-11-6/h5H,1-4H2
	TNTD	InChI = 1S/C7H7ClN2S/c8-7-9-4-6-5(10-7)2-1-3-11-6/h4H,1-3H2
Figure 10b InChI = 1S/C19H26N2O3/ C1-13-3-6-15(7-4-13) 21(2)19(23)12-24-16- 8-9-17-14(11-16)5-10 -18(22)20-17/h8-9,11, 13,15H,3-7,10,12H2,1 -2H3,(H,20,22)	ICMDT	InChI = 1S/C19H26N2O3/c1-13-3-6-15(7-4-13)21(2)19(23)12-24-16-8-9-17-14(11-16) 5-10-18(22)20-17/h8-9,11,13,15H,3-7,10,12H2,1-2H3,(H,20,22)
	ResNet101d + Transformer	InChI = 1S/C19H26N2O3/c1-13-3-6-15(7-4-13)21(2)19(23)12-24-16-8-9-17-14(11-16) 5-10-18(22)20-17/h8-9,11,13,15H,3-7,10,12H2,1-2H3,(H,20,22)
	EfficientNetb0 + Transformer	InChI = 1S/C19H26N2O3/c1-13-3-6-15(7-4-13)21(2)19(23)12-24-16-9-17-14(11-16) 5-10-18(22)20-17/h8-9,11,13,15H,3-7,10,12H2,1-2H3,(H,20,22)
	ResNet50d + RNN	InChI = 1S/C19H26N2O3/c1-13-3-6-15(7-4-13)21(2)19(23)-24-16-8-9-17(11-16) 5-10-18(22)20-17/h8-9,11,13,16H,3-7,10,13H2,1-2H3,(H,20,22)
	EfficientNetb0 + RNN	InChI = 1S/C19H26N2O3/c1-13-3-6-15(7-4-13)21(2)19(23)12-24-9-17-14(11-16) 5-10-18(22)20-17/h8-9,11,13,15H,3-7,12H2,1-2H3,(H,20,22)
	TNTD	InChI = 1S/C19H26N2O3/c1-13-3-6-15(7-4-13)21(2)19(23)12-24-16-8-9-17-14(11-16) 5-10-18(22)20-17/h8-9,11,13,14H,3-7,10,12H,1-2H3,(H,20,22)
Figure 10c InChI = 1S/C39H68O2/ c1-7-8-9-10-11-12- 13-14-15-19-37(40) 41-32-24-26-38(5) 31(28-32)20-21-33- 35-23-22-34(30(4) 18-16-17-29(2)3)39 (35,6)27-25-36(33) 38/h20,29-30,32- 36H,7-19,21-28H2, 1-6H3/t30-,32-,33+, 34-,35+,36-,38+, 39-/m1/s1	ICMDT	InChI = 1S/C39H68O2/c1-7-8-9-10-11-12-13-14-15-19 -37(40)41-32-24-26-38(5)31(28-32)20-21-33-35-23 -22-34(30(4)18-16-17-29(2)3)39(35,6)27-25-36(33) 38/h20,29-30,32-36H,7-19,21-28H2,1-6H3/t30-,32-, 33-,34-,35-,36-,38+,39+/m1/s1
	ResNet101d + Transformer	InChI = 1S/C39H68O2/c1-7-8-9-10-11-12-13-14-15-19 -37(40)41-35-24-26-38(5)35(28-32)20-25-33-35-23 -22-34(30(4)18-16-17-39(35,6)27-25-36(33)38/h20, 29-30,32-39H,7-19,21-23H2,1-6H3/t30-,32-,33+,36-,38+,39-/m1/s1
	EfficientNetb0 + Transformer	InChI = 1S/C39H68O2/c1-7-8-9-10-11-12-13-14-15-19 -46-32-24-26-33(5)31(28-32)23-21-33-35-23-22-34 (30(4)18–29(2)3)39(35,6)26-25-36(33)38/h20,29- 30,32-36H,7-19,21-28H2,1-6H3/t30-,35+,36+,38-,39-/m1/s1
	ResNet50d + RNN	InChI = 1S/C39H68O2/c1-7-8-9-10-11-12-12-12-15-19 -32-24-26-38(5)31(28-32)20-21-33-35-23-22-34(30 (4)18-16-17-29(2)3)39(35,6)27-25-36(33)38/h20, 29-30,32-36H,7-19,21-28H2,1-6H3/t30-,32-,33+,34+,35+,36-,38-,39-/m1
	EfficientNetb0 + RNN	InChI = 1S/C39H68O2/c1-7-8-9-10-11-12-13-14-15-19 -37(40)41-32-38(5)31(23-32)20-23-22-21-33-35-34 (30(4)18-16-17-29(2)3)39(35,6)27-25-36(33)38/h20 ,29-30,7-19,21-28H2,1-9H3/t30-,32-,33-,34-,38-,39-/m1/s1
	TNTD	InChI = 1S/C39H68O2/c1-7-8-9-10-11-12-13-16-15-19 -37(40)41-29-24-26-38(5)31(28-32)20-21-23-22-34 (30(4)18-18-17-29(2)3)39(35,6)27-25-36(33)38/h20, 29-30,32-36H,7-19,21-28H2,1-6H3/t30-,32-,33+,34+,35+,36-/m1/s1

As can be seen in model inference results for Figure 10b, our model is able to predict the InChI expression of this chemical structure completely and correctly. In contrast, both TNTD and some models using the transformer as a decoder show a few errors in the prediction of the hydrogen and atomic layers. In fact, such errors are common in the predictions of both types of models for other chemical structures of equal complexity, while

some models using the RNN as decoder show multiple errors in the prediction of the hydrogen layer and also the contents of the atomic layer, with the absence of certain atoms.

The other models show many errors in the prediction of chemical structures of high complexity at the atomic, hydrogen and stereochemical layers as well as large errors in the identification of atomic numbers, atomic connections and stereochemical information. "ResNet50d + RNN" does not even identify the type of stereochemical information (prefix: "s"). Our model also showed some errors in the identification of the stereochemical layers, as it did not perform well in identifying whether the atomic number of neighbors was increasing clockwise (prefix: "t") and there were errors in the discrimination of the "+/−" signs. However, the model performed significantly better than the other models in identifying other non-stereochemical layers.

At this point, the next step in our research is to improve the model's ability to recognize the stereochemical layers. One excellent method that is to use a full object detection approach, as it predicts both atom and bonds coordinates, and then identifies which atoms are connected based on the geometrical coordinates of the atoms and bonds. This additional information is fed into the model. We hope that the model itself can also restore 3D information from the 2D chemical structure without resorting to methods independent of the model and to be able to model 3D structure, so as to achieve high accuracy of detection of 3D information. This is still a very challenging process.

4.5. Inference & Fusion

We replaced the ResNet50d + RNN structure with TNTD. In the inference stage, we used the strategy of beam search (k = 16) and TTA for all five groups of models to obtain a total of 10 sets of results. If the predictions for some of the ten sets of results were consistent and met the InChI specification, these results were retained and used as the final predictions for this part of the test data. For some test data, if none of the models met the InChI specification, we continued to use beam search, set k to 64, and used a step-wise logit ensemble to fuse the five models (the output with the largest logit value in each time step was used as the input for the next time step) until the normative results were produced. The remaining results that meet the InChI specification but have disagreement were divided into categories according to the form of the results, and a voting method was used to select the final results. Specifically if the number of votes in a category was 4 or greater , it was taken as the final result . The edit distance between each result was calculated one by one, and the average of the edit distance between each result and the other results was the score of that category. The lowest score was taken as the final result. The complex reasoning and fusion process improved the results of a single model by 0.24 to 2.5 (Levenshtein distance reduction).

5. Conclusions

In this paper, we have proposed an ICMDT network for image-captioning tasks. We used a vanilla transformer-decoder as the decoder of the model, and in the encoder part, we proposed a Deep TNT block based on the TNT block that can perform a finer fusion of the local information of the image, which contains three transformer blocks that can handle image features of different levels. The internal transformer block was used to process the pixel embedding, and the pixel embedding information was projected into the small patch embedding; the middle transformer block was used to process the small patch embedding, and the small patch embedding information was fused into the large patch embedding. The exterior transformer block was used to process the large patch embedding and integrate all local information. The model models the global and pixel-level information of the image more effectively and is applicable to complex chemical structure images. We selected four representative generative models for controlled experiments, and our model had a significant advantage. In addition, we replaced the encoder with TNT, and the results were not as good as our original model. In the actual inference results, ICMDT performed much better than other comparative models in identifying non-stereochemical layers.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L.; software, Y.L.; validation, Y.L. and X.L.; formal analysis, Y.L.; data curation, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L. and X.L.; supervision, X.L. and G.C.; project administration, X.L. and G.C.; funding acquisition, X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by China University of Geosciences (Wuhan) and Xiang Li.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used to support the experiments in this paper can be accessed via the official competition link: https://www.kaggle.com/c/bms-molecular-translation/data (accessed on 1 November 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. BMS-Molecular-Translation. 2021. Available online: https://www.kaggle.com/c/bms-molecular-translation (accessed on 1 November 2021).
2. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
3. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
4. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
5. Tan, M.; Le, Q. Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
6. Wang, C.; Yang, H.; Bartz, C.; Meinel, C. Image Captioning with Deep Bidirectional LSTMs. In Proceedings of the 24th ACM International Conference on Multimedia, Amsterdam, The Netherlands, 15–19 October 2016; pp. 988–997.
7. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
8. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
9. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165.
10. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16 × 16 Words: Transformers for Image Recognition at Scale. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
11. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. Training Data-Efficient Image Transformers & Distillation Through Attention. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 10347–10357.
12. Han, K.; Xiao, A.; Wu, E.; Guo, J.; Xu, C.; Wang, Y. Transformer in Transformer. *arXiv* **2021**, arXiv:2103.00112.
13. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *arXiv* **2021**, arXiv:2103.14030.
14. Frome, A.; Corrado, G.S.; Shlens, J.; Bengio, S.; Dean, J.; Ranzato, M.; Mikolov, T. DeViSE: A Deep Visual-Semantic Embedding Model. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 2121–2129.
15. Mori, Y.; Takahashi, H.; Oka, R. Image-To-Word Transformation Based on Dividing and Vector Quantizing Images with Words. In Proceedings of the MISRM'99 First International Workshop on Multimedia Intelligent Storage and Retrieval Management, Orlando, FL, USA, 30 October 1999.
16. Kuznetsova, P.; Ordonez, V.; Berg, A.; Berg, T.; Choi, Y. Collective Generation of Natural Image Descriptions. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Jeju Island, Korea, 8–14 July 2012; pp. 359–368.
17. Elliott, D.; Keller, F. Image Description using Visual Dependency Representations. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; pp. 1292–1302.
18. Ordonez, V.; Kulkarni, G.; Berg, T. Im2Text: Describing Images Using 1 Million Captioned Photographs. *Adv. Neural Inf. Process. Syst.* **2011**, *24*, 1143–1151.
19. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *32*, 1627–1645. [CrossRef] [PubMed]

20. Mason, R.; Charniak, E. Nonparametric Method for Data-Driven Image Captioning. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Baltimore, MD, USA, 22–27 June 2014; pp. 592–598.
21. Mao, J.; Xu, W.; Yang, Y.; Wang, J.; Yuille, A.L. Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). In Proceedings of the ICLR '15 3rd International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
22. Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D. Show And Tell: A Neural Image Caption Generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3156–3164.
23. Mao, J.; Huang, J.; Toshev, A.; Camburu, O.; Yuille, A.L.; Murphy, K. Generation and Comprehension of Unambiguous Object Descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 11–20.
24. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 7–9 July 2015; pp. 2048–2057.
25. Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; Zhang, L. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6077–6086.
26. Ye, S.; Han, J.; Liu, N. Attentive Linear Transformation for Image Captioning. *IEEE Trans. Image Process.* **2018**, *27*, 5514–5524. [CrossRef] [PubMed]
27. Lu, J.; Yang, J.; Batra, D.; Parikh, D. Neural Baby Talk. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7219–7228.
28. Guo, L.; Liu, J.; Zhu, X.; Yao, P.; Lu, S.; Lu, H. Normalized and Geometry-Aware Self-Attention Network for Image Captioning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10327–10336.
29. Cornia, M.; Stefanini, M.; Baraldi, L.; Cucchiara, R. Meshed-Memory Transformer for Image Captioning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10578–10587.
30. Luo, Y.; Ji, J.; Sun, X.; Cao, L.; Wu, Y.; Huang, F.; Lin, C.W.; Ji, R. Dual-Level Collaborative Transformer for Image Captioning. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 2–9 February 2021; Volume 35, pp. 2286–2293.
31. Li, L.H.; Yatskar, M.; Yin, D.; Hsieh, C.J.; Chang, K.W. VisualBERT: A Simple and Performant Baseline for Vision and Language. *arXiv* **2019**, arXiv:1908.03557.
32. Li, G.; Duan, N.; Fang, Y.; Gong, M.; Jiang, D. Unicoder-VL: A Universal Encoder for Vision and Language by Cross-Modal Pre-training. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 2–9 February 2020; Volume 34, pp. 11336–11344.
33. Su, W.; Zhu, X.; Cao, Y.; Li, B.; Lu, L.; Wei, F.; Dai, J. VL-BERT: Pre-training of Generic Visual-Linguistic Representations. *arXiv* **2020**, arXiv:1908.08530.
34. Chen, Y.C.; Li, L.; Yu, L.; El Kholy, A.; Ahmed, F.; Gan, Z.; Cheng, Y.; Liu, J. UNITER: UNiversal Image-TExt Representation Learning. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 104–120.
35. Zhou, L.; Palangi, H.; Zhang, L.; Hu, H.; Corso, J.; Gao, J. Unified Vision-Language Pre-Training for Image Captioning and VQA. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 2–9 February 2020; Volume 34, pp. 13041–13049.
36. Lu, J.; Batra, D.; Parikh, D.; Lee, S. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. *arXiv* **2019**, arXiv:1908.02265.
37. Tan, H.; Bansal, M. *LXMERT: Learning Cross-Modality Encoder Representations from Transformers*; Inui, K., Jiang, J., Ng, V., Wan, X., Eds.; EMNLP/IJCNLP (1); Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 5099–5110.
38. Hu, J.; Shen, L.; Sun, G. Squeeze-and-Excitation Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
39. Li, X.; Wang, W.; Hu, X.; Yang, J. Selective Kernel Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 18–20 June 2019; pp. 510–519.
40. Zhang, H.; Wu, C.; Zhang, Z.; Zhu, Y.; Zhang, Z.; Lin, H.; Sun, Y.; He, T.; Mueller, J.; Manmatha, R.; et al. ResNeSt: Split-Attention Networks. *arXiv* **2020**, arXiv:2004.08955.
41. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-Local Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7794–7803.
42. Yin, M.; Yao, Z.; Cao, Y.; Li, X.; Zhang, Z.; Lin, S.; Hu, H. Disentangled Non-Local Neural Networks. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 191–207.
43. Yuan, Y.; Huang, L.; Guo, J.; Zhang, C.; Chen, X.; Wang, J. OCNet: Object Context Network for Scene Parsing. *arXiv* **2018**, arXiv:1809.00916.
44. Hu, H.; Gu, J.; Zhang, Z.; Dai, J.; Wei, Y. Relation Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 3588–3597.
45. Beal, J.; Kim, E.; Tzeng, E.; Park, D.H.; Zhai, A.; Kislyuk, D. Toward Transformer-Based Object Detection. *arXiv* **2020**, arXiv:2012.09958.

46. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable DETR: Deformable Transformers for End-to-End Object Detection. *arXiv* **2020**, arXiv:2010.04159.
47. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 213–229.
48. Zheng, S.; Lu, J.; Zhao, H.; Zhu, X.; Luo, Z.; Wang, Y.; Fu, Y.; Feng, J.; Xiang, T.; Torr, P.H.; et al. Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; pp. 6881–6890.
49. Chen, H.; Wang, Y.; Guo, T.; Xu, C.; Deng, Y.; Liu, Z.; Ma, S.; Xu, C.; Xu, C.; Gao, W. Pre-Trained Image Processing Transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; pp. 12299–12310.
50. Hendrycks, D.; Gimpel, K. Gaussian Error Linear Units (GELUs). *arXiv* **2016**, arXiv:1606.08415.
51. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional Sequence to Sequence Learning. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 1243–1252.
52. Albumentations. 2021. Available online: https://github.com/albumentations-team/albumentations (accessed on 1 November 2021).
53. Zhang, M.R.; Lucas, J.; Ba, J.; Hinton, G.E. Lookahead Optimizer: k steps forward, 1 step back. In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019; pp. 9593–9604.
54. Liu, L.; Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; Han, J. On the Variance of the Adaptive Learning Rate and Beyond. *arXiv* **2019**, arXiv:1908.03265.
55. Raunak, V.; Dalmia, S.; Gupta, V.; Metze, F. On Long-Tailed Phenomena in Neural Machine Translation. In Proceedings of the EMNLP (Findings), Association for Computational Linguistics, Online, 16–20 November 2020; pp. 3088–3095.
56. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
57. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
58. Touvron, H.; Vedaldi, A.; Douze, M.; Jegou, H. Fixing the Train-Test Resolution Discrepancy. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8252–8262.
59. RDKit. 2021. Available online: https://github.com/rdkit/rdkit (accessed on 1 November 2021).
60. Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; He, K. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv* **2017**, arXiv:1706.02677.