# SwimNet: A Shifting Window Transformer for Translating Written Mathematical Expressions

Charles Immendorf
University of Washington
chazi [at] uw [dot] edu

Liam Gene Ping Chu
University of Washington
liamchu [at] uw [dot] edu

Davin Seju
University of Washington
dseju [at] uw [dot] edu

## 1. Abstract

We present SwimNet: a shifting window transformer encoder-decoder model for translating images of mathematical expressions to LaTeX code. With the emergence of deep learning as an automated translator, we design and implement a transformer-based neural network architecture to tackle this new challenge in OCR. We use the Im2Latex dataset, which is utilized in seven publications – none of which implement transformer-based models. Although we do not outperform the state of the art attention-based model under this dataset, we show that with careful refinement of the model, and longer training times, a transformer-based architecture is a strong competitor for resolving challenges involving mathematical expression recognition and translation.

## 2. Introduction

Automatically translating handwritten symbols into text has gained significant traction in the past few years with the popularization of deep learning models. In particular, much work has been done on segmenting handwritten documents (of general text, not just math) into individual lines; converting an image of math symbols into the corresponding LaTeX; and classifying math symbols from data of individual handwritten and printed math symbols. We analyze the performance of a transformer neural network and assess its viability in the Printed Mathematical Expression Recognition (PMER) field.

### 2.1. State of related work

Encoder/decoder deep learning models are popularly used for recognition and classification purposes in the handwritten mathematical expression recognition (HMER) and PMER fields. They do not require any grammar rules to recognize expressions, which avoids the need for semantic structural analysis which is common in grammar based models – another popular tool for expression recognition (Garkal et. al.). These deep learning encoder/decoder models use an encoder to extract high level features and then decode results into LaTeX formulas (the encoder and decoder trained at the same time which speeds up overall training time).

The researchers also note that they make use of a new CNN technique referred to as DenseNet. Densely connected convolutional neural networks (DenseNet) have more connections for a given number of layers than standard CNNs by connecting all layers directly with each other. Then, each layer only adds a small set of feature-maps to the "collective knowledge" of the network, and passess the rest of the information along unchanged. This helps to preserve information that might have been lost in the earlier stages of the network, which helps to alleviate the issue of vanishing gradients, strengthens feature propagation, encourages feature reuse, and reduces the number of parameters (Huang et. al.).

Other papers work with other types of models to convert images to LaTeX such as using attention, LSTMs, and unsupervised learning. LSTMs have better performance than CNN based models but the accuracy difference is not significantly better to lose the parallelization and performance of the CNNs (Genthial et. al.). The problem can be considered to be a version of image classification and captioning. In this space CNNs also perform well even compared to LSTM and Attention models (Ghandi et. al.).

### 2.2. Problem and Insight

Some of this work had only dealt with images of text of the math symbols, rather than handwritten math symbols. From our literature review, we found that some models trained on non-handwritten math symbols may not perform well on accurately identifying handwritten symbols.
Also, the usage of the transformer architecture hadn't been used with the given related work. Given the success of other models We believe that that architecture may perform better than LSTMs, particularly in the task of decoding encoded images of math symbols into their LaTeX symbols.

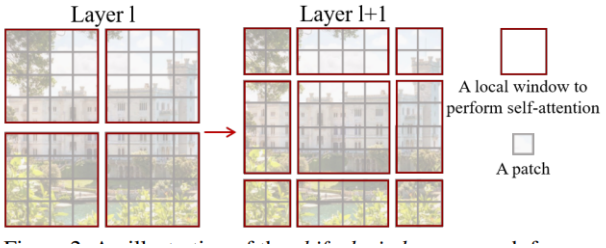


Figure 1. An illustration of how a swin transformer uses a shifted window method to focus attention on parts of an image

## 3. Related Work

In this section, we detail a brief summary of recent work with transformer models, distance metrics, PMER, and encoder-decoder architectures.

### 3.1. Transformer Neural Networks

Transformers have shown great success in NLP models over previous methods such as RNNs, CNNs, and LSTMs. Using several self-attention and multi-headed attention layers, transformers are able to understand dependency relationships in long text strings without losing as much accuracy as an LSTM would (Liu et. al.). Only recently, machine learning scientists have started expanding the use of transformers to other inputs and tasks such as image captioning, Since transformers not only pay attention to its input but also considers previous outputs as well, they have been used in encoder-decoder models over other architectures for image captioning. However CNNs are still commonly used in these tasks to encode features of an image to pass into another transformer encoder (Wang et. al.).

But, Dosovitskiy et. al. had used transformers to also encode features the image, by splitting the image into a grid of fixed-sized patches and feeding those into the transformer encoder. Their Vision Transformer (ViT) achieved state-of-the-art on various image recognition tasks (Dosovitskiy et. al.).

Then, Liu et. al. created the Swin Transformer which, unlike the ViT, which performed self-attention on all patches of the image at once, instead did self-attention on windows of patches of the image, which is more computationally efficient. Deeper into the model, these windows are merged to make larger windows, which encode larger parts of the image. The Swin Transformer also used shifted windows (figure 1) which can encode information between the windows of the previous layer (Liu et. al.). With the Swin Transformers, Wang et. al. were able use multiple transformers without CNNs to achieve new state of the art image captioning (Wang et. al.).

### 3.2. Distance Metrics

A difficulty faced by many engineers is finding a suitable way to express how ”unhappy” they are with an incorrect result compared to the ground truth in an attempt to help the learning system achieve a lower loss. Popular methods for determining a resulting sequence’s distance from the ground truth include simpler methods like the Hamming distance to more complex methods such as the aggregated cross-entropy (ACE) counting based loss (Xie et. al). Interestingly, in their research on HMER, Li et. al. employed the use of a hybrid loss function, that took the overall loss on the entire expression and combined that with the loss on individual symbols.

### 3.3. Printed Mathematical Expression Recognition

In the past few years, many advancements in the field of PMER have been made. In particular, a refined process for analyzing a given mathematical expression is detailed in Chaudhari et. al.’s research paper. They describe how a mathematical expression is recognized in these steps: ”Recognizing a single mathematical symbol; recognizing the relation between all symbols by using their traces on the interface; Converting the symbols into LaTeX and arranging them to form complete mathematical expressions” (Chaudhari et. al). Alvaro et. al. condense this further into two high level ideas: symbol recognition and structural analysis.

In their process, Alvaro et. al. described how they segmented the mathematical expression (ME) into symbols by computing connected components. Then, a symbol classifier translates individual symbols. However, the researchers note that with just a one-to-one conversion approach, they faced two problems: ”First, the system currently is not able to deal with touching characters. Second, an important problem was the multiple connected components symbol detection” (Avlaro et. al.). To address this issue, they implement structural analysis, where they compute probabilities of potential symbols for each type of spatial relation based on geometric features (center, horizontal offset, vertical offeset, etc.).

### 3.4. Encoder-Decoder Architecture with Symbol-Level Features

The paper Encoder-Decoder Architecture with Symbol-Level Features by Fu et. al strives to solve a similar problem except with already rendered LaTeX. The architecture contrasts with past work as it tries to encode an image through symbol blocks. Each symbol block is not a character but can be part of a character with the idea being that CNN will be able to encode the features of each block and another fully connected layer can encode the positions. Once these features are encoded, the positions are used to calculate an attention score using positional correction attention. These scores are finally fed into a transformer

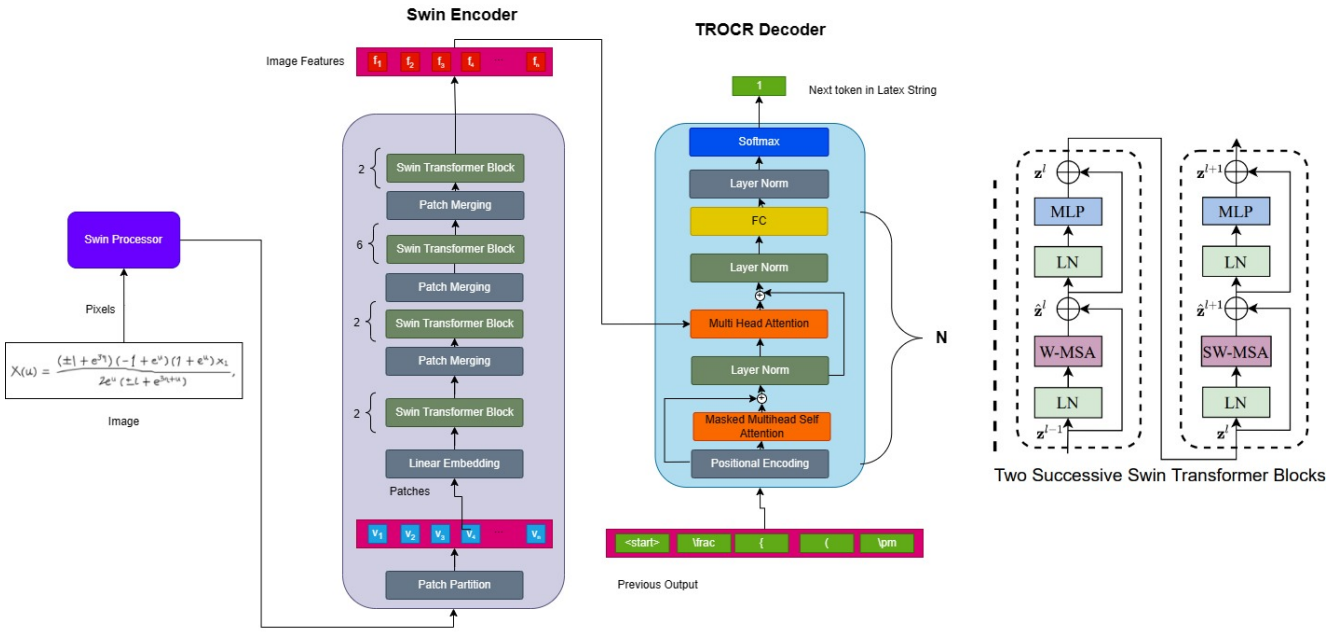


Figure 2. SwimNet Transformer Encoder Decoder architecture with Swin Encoder

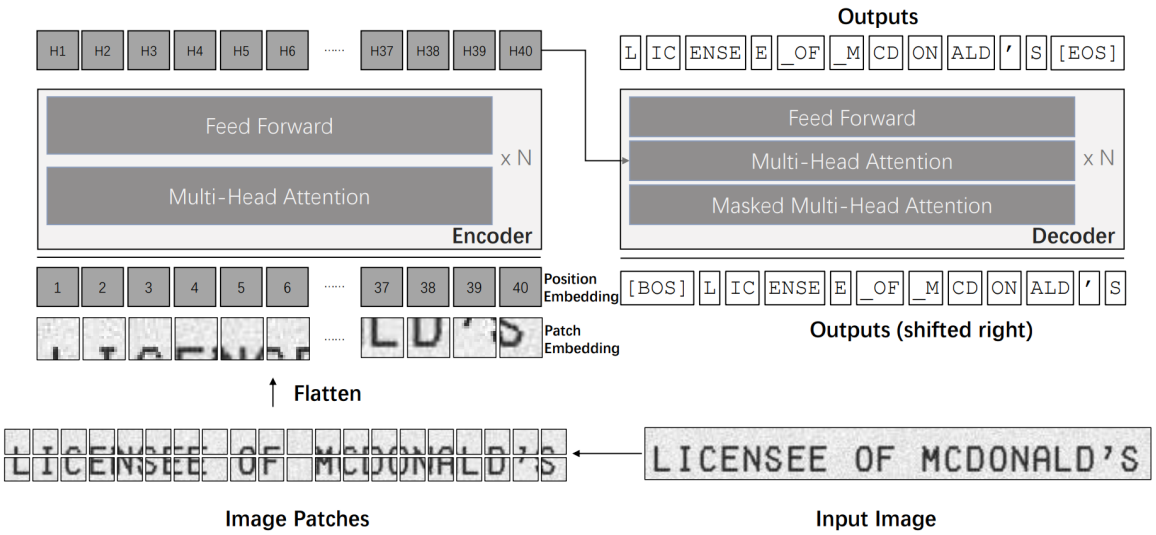


Figure 3. TrOCR transformer encoder decoder model architecture

that outputs the final LaTeX string. The authors found that this model was able to perform better than previous models using LSTMs and CNNs and regular image captioning encoder decoder architectures getting a minimum of 88 percent (Fu et. al). This model is however limited in that it deals with non-handwritten LaTeX which is more standardized.

## 4. Methods

In this section, we discuss our methodology and approach to tackling the challenge of translating mathematical text within images to LaTeX formulas. We first identify our dataset and then transition to explaining our overall model SwimNet (Shifted Window Math Network). Next, we identify the relevant architectures employed within the pipeline. Finally, we detail our method for determining loss and the

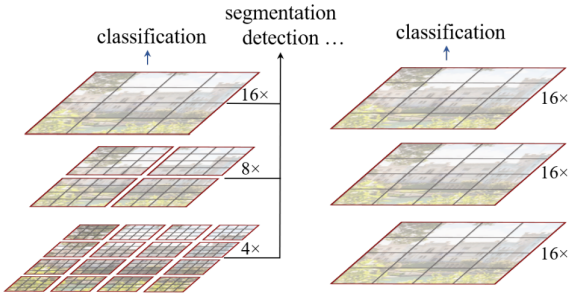


Figure 4. A comparison between Swin Windows on the left and a ViT encoder

distance between incorrect mathematical translations from the ground truth.

## 4.1. Dataset

In this experiment, we utilize the Im2Latex-100K dataset which contains "a large collection of rendered real-world mathematical expressions collected from published articles" (Deng et. al.). This provides us with about 100,000 formulas to aid in the process of training, validating, and testing our model. Specifically, we used their handwritten version, which is a synthetic dataset where the printed characters in the images are replaced with images of handwritten, randomly selected, corresponding characters. Each formula is represented on a PNG image of fixed size, with the formula in grayscale, and the background white after preprocessing. A LaTeX markup is provided with each image to support the ground truth.

## 4.2. Model

Our SwimNet architectures attempts to solve the handwritten mathematics to LaTeX problem relying on a simple encoder decoder architecture relying on Transformers at every stage. The architecture though relatively simple can be expanded on with more transformer layers. We also trained the data set on TrOCR another transformer encoder decoder model already pre-trained for regular handwritten text.

### 4.2.1 TrOCR

TrOCR is the state of the art OCR model trained on the IAM handwritten dataset acheiving about 96 percent accuracy. Unlike other OCR models TrOCR doesn't use a CNN to extract image features and instead relies on a ViT (for the base version of the model) or DeiT (for the small version of the model) encoder that takes image patches and feeds it into a transformer directly. A DeiT (Data-efficient image Transformer) is a version of ViT that is trained more efficiently in terms of time and the amount of data needed for sufficient training (Touvron et al.). Once the ViT/DeiT encodes the image features TrOCR then passes them to a decoder initialized with RoBERTa which is a language decoder model. Using previous outputs and image features the encoder decoder predicts the next word in the sequence as demonstrated in figure 3 (Li et.al.).

### 4.2.2 Swin Encoder

Unlike other models where images are encoded with a CNN, SwimNet relies on the Swin-Encoder to extract features from the image. This step also replaces symbol level extractions of other papers due to the inherent patch separation of transformers compared to the pixel level convolutions of a CNN. At this step in the process we hypothesize that the symbol level features will be extracted along with positional data on each symbol in relation to others due to the self-attention modules in the Swin-Encoder.This contrasts slightly with the approach in TrOCR which uses a ViT (or DeiT) style encoder. As shown in figure 4 and 1 the Swin encoder performs self-attention on different size windows to detect connections between patches thus potentially improving spatial inference of our mathematical equations (Wang et. al.).

### 4.2.3 Decoder

This final transformer takes the image features and previous outputs to produce the next token. Using attention on previous outputs and the encoded input the decoder should be able to produce the next character in the sequence. An initial input character will be given such as <s> to indicate the first output should be based off of the encoded features and an </s> character will be the final output the indicate that the whole LaTeX string was generated. Other special tokens include <unk> for unknown tokens in our data set and <pad> to ensure all output strings are the same length. The decoder is initialized with the TrOCR decoder pre-trained on the IAM data set. The input words are passed initially through a positional encoder and then into a Masked MSA layer which will help the model pay attention to only the previous words and not future ones. Once passed through a layer norm it joins with the encoded input in a Multi-Head attention layer which will help the model compare previous outputs with the image to focus on the next output. The output will then go through a layer norm and a fully connected layer. After N blocks of these modules the final output is soft maxed to provide the symbol with the largest probability. This final symbol is added to the LaTeX string and passed back into the decoder until an </s> character is generated. Using beam search the decoder outputs multiple possible strings to select the highest confidence output.

### 4.3. Loss Function

When training, we quantify the severity of an incorrect answer in comparison to the ground truth with a masked language modeling loss. Essentially, a masked language model (MLM) predicts missing (masked) words in a given sentence. It generates its prediction based on the context of the surrounding words. It's loss function is a cross entropy loss between predicted probabilities (logits) and labels. In our case, we apply the MLM loss function to help our network better understand the input tokens (LaTeX characters) within the context of the expression.

## 5. Experiments

In this section, we will first introduce our method for evaluating our model as well as baseline comparisons. Next, we discuss our implementation details and design, and finish with a report on performance and results.

### 5.1. Evaluation Criterion and Baselines

The performance metric we employ in this experiment to help assess the accuracy of our system is the Word Error Rate (WER). WER penalizes based on entire words as opposed to characters. The error rate is calculated as the total number of word errors divided by the number of words in the correct sentence. More specifically, the number of errors is the sum number of word substitutions, word insertions, and word deletions required to turn the predicted sentence into the correct sentence (Morris et al.). The accuracy could then be thought of as the complement to the WER (i.e., $1 - \text{WER}$). Note that the WER, in practice, can return an error rate of more than 100%. This is due to the WER scaling on number of words for given number of errors. For example, consider the case where the true output had 10 words, but we generated 20 words. Suppose all words generated were incorrect, then we have $\frac{errors}{words} = \frac{20}{10}$, which evaluates to a 200% error rate.

Since we model the generation of our mathematical expressions with tokens expressed at a "word" level (for instance, $\sin\theta$), we provide a brief description of a word level model. A word level model compiles a list of every possible word generated in order to help calculate the probabilities of each word being generated under a given context. This could potentially be a very large compute in the hidden layers, but tends to make more sense contextually due to the model having guaranteed complete words. In their tutorial on evaluation metrics for automatic speech recognition (ASR), HuggingFace noted that WER is the preferred metric in natural language processing (NLP).

For comparison, a character level evaluation tends to provide higher accuracy readings (compared to a word level model) due to only needing to get individual characters correct. However, the resulting output often makes less sense contextually as the model needs to correctly interpret every individual token, which allows for higher variation. Fortunately, the parameter space tends to be smaller since the training vocabulary is a fixed size, which limits the size of the weight matrices.

As a baseline for model comparisons, we note that there have only been seven publications using the Im2Latex-100k data set, none of which implement the use of of a transformer based model. Under this consideration, the state of the art under a similar attention-based architecture is Wang et. al.'s CNN encoder with LSTM decoder. Under this model, they achieve an 82.33% word accuracy.

### 5.2. Implementation Details

Before training the models, the training/validation/test sets were preprocessed and filtered with the help of scripts by Deng et al., the creators of the handwritten Im2Latex dataset. Examples whose formulas had more than 150 tokens were removed, and examples whose images exceeded 500 pixels in width or 150 pixels in height were removed. The resulting train, validation, and test sets had 68258, 8547, and 8557 examples respectively.

For our training, we decided to experiment with three different encoder-decoder models. The first is the architecture described earlier in the architecture section, with Swin Transformer's encoder ("microsoft/swin-small-patch4-window7-224") and TrOCR's decoder ("microsoft/trocr-base-handwritten"). The second uses both TrOCR's encoder and decoder, where the TrOCR encoder is a ViT that was trained on text, rather than a Swin transformer that was trained on general images. The third uses TrOCR's small encoder and TrOCR's small decoder ("microsoft/trocr-small-handwritten"), where the encoder is a DeiT. These pretrained models were obtained from Hugging Face Transformers and finetuned on the filtered Im2Latex dataset.

During inference, we had to use a small subset of the validation set during evaluations for the TrOCR models, since evaluations took a significantly long time (sometimes longer than 1 training epoch) to perform. A size-1000, randomly selected (without replacement) subset of the validation set was used. The subset used was kept the same. But the full test set was used for all evaluations on the test set.

### 5.3. Performance and Results

The Swin-encoder-TrOCR-decoder model was trained for 10 epochs, which took about 11 hours on an NVIDIA T4 GPU. It achieved a WER of 0.540 on the test set. It was trained using a learning rate of 5e-6, a weight decay of 1e-4, and a training batch size of 15.

The TrOCR-encoder-decoder model was trained for 15 epochs, which took about 15 hours on an NVIDIA V100 GPU. It was trained using a learning rate of 5e-7 for 6


Figure 5. An example of an output early on in our training process which showcases the model's ability to learn the structure of LaTeX code (this image in particular had 10% accuracy due to the model not being able to fill in the curly braces).

epochs then 1e-7 for 9 epochs, a weight decay of 1e-4, and a training batch size of 8. It achieved a WER of 0.459 on the test set. This was the best-performing model of the three.

The small TrOCR-encoder-decoder model was trained for 5 epochs, which took about 2.5 hours on an NVIDIA V100 GPU. It was trained using a learning rate of 5e-6, a weight decay of 1e-4, and a training batch size of 28. It achieved a WER of 0.702 on the validation set.

For all three models, an AdamW optimizer was used for training. And beam search with a beam size of 4 was used during inference for all three models.

## 6. Discussion

Although our model does not outperform Wang et. al.'s CNN encoder with LSTM decoder, we have a strong start, and ample evidence, to show that a transformer-based model is a good solution for PMER/HMER. In particular, our model shows a strong ability to recognize grammatical structure of a mathematical expression. Although some of the tokens may not be correct, the LaTeX formulas are similar in grammatical structure. Additionally, the model appropriately encapsulates certain input tokens with { and } (see Figure 5).

In terms of areas where we seek to improve, we note that Wang et al. trained their model for 23 hours over 16 epochs, while we only did 11 hours of training over 10 epochs for the Swin-encoder-TrOCR-decoder, and 15 hours of training over 15 epochs for the TrOCR-encoder-decoder. We hypothesize that our model would have higher performance if we could train our model for longer. We also would like to see how well each model would've performed under the same amount of time as Wang et al., so that would be considered in future work. Additionally, we considered implementing a custom loss function which penalizes based on the token generated, and the scales that token on how relevant it was mathematically to the scheme. In the future, we would be interested in experimenting with this method.

## References

[1] A. Garkal, A. Pal, and K. P. Singh, "HMER-Image To LaTeX: A Variational Dropout Approach," in 2021 5th Conference on Information and Communication Technology (CICT), Dec. 2021, pp. 1–5. doi: 10.1109/CICT53865.2020.9672359.

[2] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul. 2017, pp. 2261–2269. doi: 10.1109/CVPR.2017.243.

[3] Taraneh Ghandi, Hamidreza Pourreza, and Hamidreza Mah- yar. Deep learning approaches on image captioning: A review. ACM Computing Surveys, 56(3):1–39, oct 2023.

[4] Guillaume Genthial. Image to latex. 2017

[5] Martin Thoma. The hasyv2 dataset, 2017

[6] Islam, A., Anjum, T. & Khan, N. Line extraction in handwritten documents via instance segmentation. IJDAR 26, 335–346 (2023). https://doi.org/10.1007/s10032-023-00438-7

[7] P. Chaudhari, B. Prajapati, and P. Puvar, "Handwritten Mathematical Equations Conversion to LaTeX Equivalent," IJCTT, vol. 68, no. 4, pp. 248–252, Apr. 2020, doi: 10.14445/22312803/IJCTT-V68I4P138.

[8] F. Alvaro, J.-A. S´nchez, and J.-M. Benedi, "Recognition of Printed Mathematical Expressions Using Two-Dimensional Stochastic Context-Free Grammars," in 2011 International Conference on Document Analysis and Recognition, Beijing, China: IEEE, Sep. 2011, pp. 1225–1229. doi: 10.1109/ICDAR.2011.247.

[9] B. Li et al., "When Counting Meets HMER: Counting-Aware Network for Handwritten Mathematical Expression Recognition." arXiv, Jul. 23, 2022. Accessed: Nov. 15, 2023. [Online]. Available: http://arxiv.org/abs/2207.11463

[10] Fu, Y., Liu, T., Gao, M., & Zhou, A. (2020). EDSL: An encoder-decoder architecture with symbol-level features for printed mathematical expression recognition. https://doi.org/10.48550/ARXIV.2007.02517

[11] "Hamming Distance - an overview — ScienceDirect Topics." Accessed: Nov. 15, 2023. [Online]. Available: https://www.sciencedirect.com/topics/engineering/hamming-distance

[12] A. Morris, V. Maier, & P. Green. (2004). From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. In: INTERSPEECH 2004 - ICSLP, 8th International Conference on Spoken Language Processing. DOI:10.21437/Interspeech.2004-668

[13] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, & H. Jegou. (2021). Training data-efficient image transformers & distillation through attention. In: Proceedings of the 38th International Conference on Machine Learning. Accessed: Dec 11, 2023. [Online]. Available: https://proceedings.mlr.press/v139/touvron21a.html

[14] Xie, Z., Huang, Y., Zhu, Y., Jin, L., Liu, Y., Xie, L.: Aggregation cross-entropy for sequence recognition. In:

Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition. pp. 6538–6547 (2019)

[15] B. Li et al., “When Counting Meets HMER: Counting-Aware Network for Handwritten Mathematical Expression Recognition.” arXiv, Jul. 23, 2022. Accessed: Nov. 15, 2023. [Online]. Available: http://arxiv.org/abs/2207.11463

[16] Wang, Y., Xu, J., & Sun, Y. (2022). End-to-end Transformer based model for image captioning. https://doi.org/10.48550/ARXIV.2203.15350

[17] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin Transformer: Hierarchical vision Transformer using shifted windows. https://doi.org/10.48550/ARXIV.2103.14030

[18] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, “Image-to-Markup Generation with Coarse-to-Fine Attention.” arXiv, Jun. 13, 2017. Accessed: Nov. 16, 2023. [Online]. Available: http://arxiv.org/abs/1609.04938

[19] Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., & Wei, F. (2021). TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models (Version 5). arXiv. https://doi.org/10.48550/ARXIV.2109.10282

[20] Z. Wang and J.-C. Liu, “Translating Math Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-level Training.” arXiv, Sep. 09, 2019. Accessed: Dec. 09, 2023. [Online]. Available: http://arxiv.org/abs/1908.11415