Long Sequences Processing with Hierarchical Shifted Window Attention

Anonymous ACL submission

Abstract

Long sequences processing has been a challenging task for traditional transformers as the full self-attention mechanism scales quadratically with respect to the input sequence length. Different designs over attention have been applied to address this issue to reduce computational complexity and take into account both global and local features of the document. Here, we describe an approach to use shifted windows in attention calculation to construct a hierarchical structure in processing long sequences. Our work is inspired by the unprecedented performance of Swin Transformer in Computer Vision tasks. Hierarchical structure reduces computational complexity and improves training and inference speed, while shifted window attention enables communication across windows. Our Hierarchical Shifted Window (HSW) method has achieved the highest classification F1 score in almost all the categories in Arxiv Dataset among baseline models and has reduced the training and inference speed by 10 times compared with Longformer. We hope our work could shed some light on unified modeling in both CV and NLP, as Swin Transformer shows great performances in both fields.

1 Introduction

The use of transformer models has been highly prevalent nowadays in Natural Language Processing (NLP) domain as well as Computer Vision (CV). However, for both fields, the high computational complexity of the full self-attention mechanism has been a major problem. For NLP, since language is highly context-based, the necessity to conduct self-attention on every single word has been up to debate; for CV, given the much higher resolution of pixels, the computational complexity for images requires too much resource that many researchers don't have access to. To solve this problem, in CV, a method called Shifted Window (Swin) Transformer (Liu et al., 2021) only computes local self-attention within each non-overlapping window partition on the image, and make use of hierarchical structure and patch merging to extract detailed information. Swin Transformer also connects the windows together through the clever use of shifted windows that boosts its ability to model long range dependency. With such a high-performing model in the field of CV, we want to see whether it can be applied in NLP as well. With its clear application to CV, this method is valuable for the field of NLP as well because it can largely reduce computational complexity especially for the heavy computing tasks such as long sequences processing. The most challenging part would be to find a way to design shifted windows in a fashion that can be applied to languages. If this can be achieved, it would not only reduce computational complexity for NLP models, but also greatly promote unified modeling in the prominent fields of NLP and CV.

2 Related Works and Motivations

Recently, attention-based transformer models (Ashish et al., 2017) have become the state of the art in both Computer Vision and Natural Language Processing, due to its ability to model long range dependency and reduce inductive bias introduced in traditional approaches. The single-head attention layer is defined as:

$$Attention(Q, K, V) = \sigma(\frac{QK^T}{\sqrt{d_k}})V$$

where Q, K, V are query, key and value respectively, and σ is a scoring function (e.g. softmax or hardmax). Transformer model is made up of 12 multi-head self-attention layers.

2.1 Transformer on Long Sequence Processing

The main challenge for attention-based transformer models is that the computational complexity of

self-attention layer is quadratic with respective to the input size, which limits its ability to process very long sequences, due to the computational and memory constraint in hardware. Therefore, how to reduce computational cost while maintaining a comparatively good accuracy has been a popular research topic. Currently, there have been multiple innovative methods targeting this problem, with most of them modifying the attention mechanism to reduce the computation from quadratic complexity to linear complexity.

2.1.1 Longformer

Longformer (Beltagy et al., 2020) aims to reduce the computation complexity of BERT-like transformers to handle long sequences. Given a sequence length n, instead of using full n^2 attention, it substitutes it with sliding window attention and several global attention tokens. With a fixed window size w, on each side, each token attends to $\frac{1}{2}w$ tokens, which results in a computation complexity of $O(nw)$. In order to learn task-specific representations, global attentions that attend to all tokens across the entire sequence are used. This architecture does not add computation complexity because the number of tokens is small relative to and independent of the sequence length n, leaving the complexity of the combined local and global attention still $O(n)$. Although sliding window attention mechanism can reduce the computation complexity, its performance is compromised as the word dependency in a sentence might cross a long range.

2.1.2 Big Bird

The BigBird approach (Zaheer et al., 2020) is inspired by the idea of Longformer, but it combines Longformer with random attention, which is calculated as self-attention from randomly chosen tokens. Specifically, with a width of w for self-attention, queries in BigBird attends to r random keys. Each query attends to $\frac{1}{2}w$ tokens to the left of its location and $\frac{1}{2}w$ to the right, and g tokens are made globally from either existing tokens or extra added tokens. By employing a sparse, rather than full, attention mechanism, this method reduces the quadratic dependency on sequence length n to linear, and enlarges the communication range compared with sliding window attention. However, in order for a word to see all the other words, this method needs to increase the number of layers inside the transformer, making it computationally more complicated.

2.1.3 Multi-scale Transformer

The multi-scale transformer (Subramanian et al., 2020), whose intuition is different from that of Longformer and Big Bird, doesn't aim to modify the attention mechanism. Instead, it adopts the idea of learning language representation at multiple scales and proposes variants of hierarchical structure that are able to learn representation of large text corpus with less computation complexity than traditional Transformer. It proposes three multi-scale transformer architectures, Top-down, Bottom-up, and Retina model. The Top-down model builds representations from coarse to fine scales; the Bottom-up model aggregates representations from different scales before operating at the finest scale while building representations from fine to coarse; the Retina model treats nearby information as fine-grained and distant information as coarse. The bottom-up hierarchical structure proposed in this paper is similar to Swin Transformer in subsection 2.2; however, instead of the shifted-window self-attention mechanism, this paper simply designed attention masks to reduce the computation complexity. Therefore, we believe it is possible that applying the shifted-window mechanism can further improve the performance of multiple scale language model on learning long-sequence text.

2.2 Motivation: Swin Transformer in CV

Swin Transformer (Liu et al., 2021) is the current SOTA backbone for more than 16 vision tasks such as object detection and semantic segmentation. Its intuition is to combine the locality property of CNN with transformer, while maintaining the ability to model long range dependency through shifted window that enables communication between image patches. Swin Transformer computes self-attention locally within non-overlapping windows that partition an image with fixed the number of patches in each window, therefore successfully achieving linear computational complexity relative to the image size. This paper has greatly inspired us to apply shifted window transformer to long text processing. We believe that for a long sequence input, the inductive bias of locality is also helpful, as paragraphs separated far away generally have lower dependency compared to those located together. Besides, we also believe that, for less generative tasks such as text classification and summarization, high-level (such as sentence-level or paragraph-level) seman-

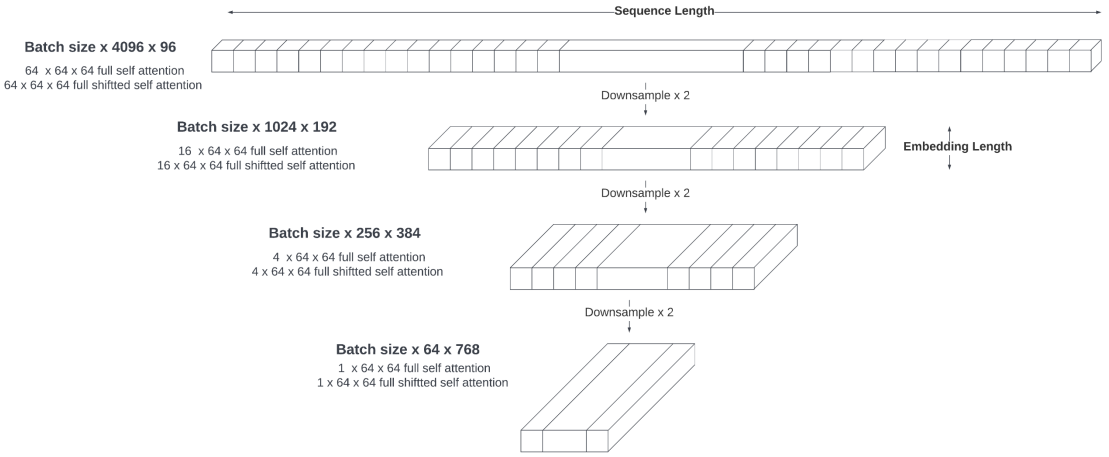


Figure 1: Hierarchical structure of our proposed method

tic representation is more useful than low-level (such as word-level). Therefore, we see the opportunity of applying hierarchical shifted window attention to long sequence processing, to reduce the computation burden, while improving the accuracy of sliding window attention approach by allowing communication between adjacent blocks.

3 Methodology

Here, we present Hierarchical Shifted Window (HSW) model to solve this problem. Our approach has two folds. First, we proposed a **hierarchical structure** for faster training and inference process and reduced computational requirements. Figure 1 shows the hierarchical structure. In the first layer, each of the 4096 words is treated as token of size 96. Patch merging is used to down-sample the feature size from layer 1 to layer 2. After three times of down-sampling, the final size of the long sequence representation has become 64×768. In comparison, the original transformer model and the Longformer model have a final representation of size 4096×768. Therefore, our hierarchical structure greatly reduces training and inference time by using only 64 tokens to summarize the sequence. Additionally, after each down-sample operation, the tokens in the next layer get to a higher level representation. The lower level of our hierarchical structure extracts word/sentence level information, and the higher level attentions extract section/article level information. The pyramid representation of language information is more useful for less generative tasks, including text classification.

Second, we proposed the use of **shifted window attention** in long sequence processing to enable communication across attention windows. The previously proposed window attention methods are effective at reducing computational complexity, but they limit the transformer's ability to model long-range dependencies. We adapted the idea of shifted window attention from the Swin Transformer model for Computer Vision, where the attention is shifted to the right by half of the window length. Through this shifted attention mechanism, tokens within a certain attention window can perform attention with tokens in the next window, as shown in Figure 2. In our implementation, for layer 1, the first 64 tokens are concatenated with the last 64 tokens to perform self-attention, maintaining the computational intensity before and after window shifting. In order to prevent the first tokens from performing attention with the last tokens, an attention mask is added to the self-attention matrix, as stated in the Swin Transformer paper (Liu et al., 2021).

Similar to the structure of Swin transformer, our proposed model has two transformer layers. For each layer, there are two self-attention stages, where the first stage performs the original window attention and the second stage performs shifted window attention. After each pair of transformer blocks, the length of the representations is down-sampled by a factor of four and the feature size is

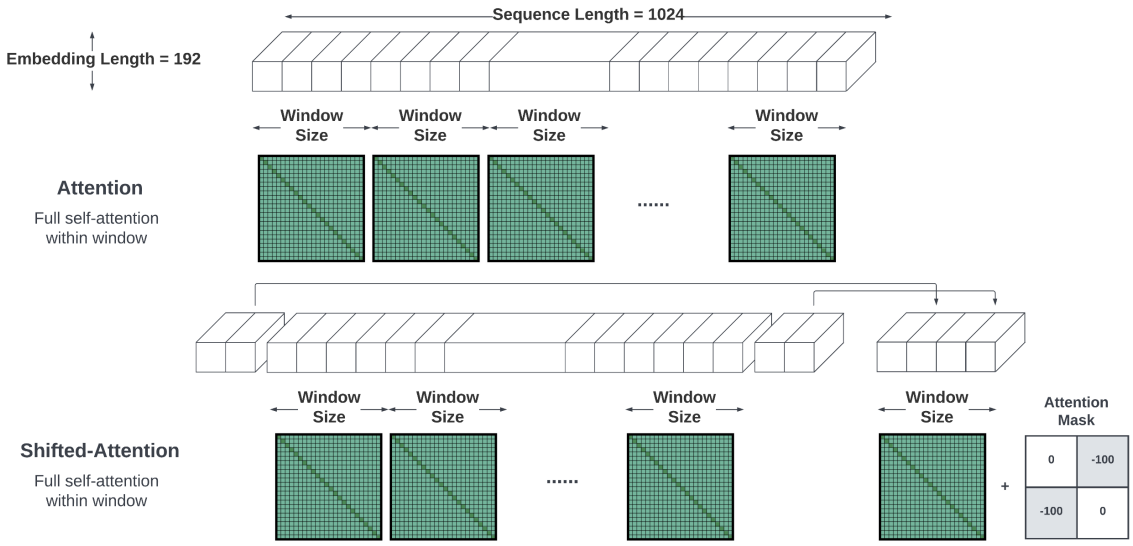


Figure 2: Shifted Window Attention and the Attention Mask

increased by a factor of two. A classification head is added on top of the final representations of size 64×768.

4 Results

In this section, we will present the performances of the baseline models and Hierarchical Shifted Window (HSW) model for a multi-class classification task. Specifically, we will be comparing the results of our HSW model with two baseline models using Longformer (Beltagy et al., 2020).

4.1 Dataset and Experimental Setup

We used the Arxiv Classification Dataset (He et al., 2019) in this project. This Dataset is created specifically for the purpose of long-text classification with 11 scientific categories and each category contains around 2500 papers. Each sample in the Dataset has over 4000 tokens in length. For pre-trained Longformer model, we randomly selected 500 samples per topic as training data, 100 samples per topic as validation data, and 100 samples per topic as test data. The samples are first shuffled and then split to make sure there is no chronological bias across the train, validation, and test group. We built a customized torch Dataset class to load the text data and their corresponding labels. The pre-trained `roberta-base` tokenizer is used to encode the word tokens, following the procedure in the original Longformer paper (Beltagy et al., 2020). For Longformer model without pre-trained weights, 2300 samples per category are chosen to be the training data. Due to the the limited computational resources and the low training speed of Longformer, pre-trained Longformer model is trained for 5 epochs (around 30 hours) while the no pre-trained Longformer model is trained for 10 epochs (around 60 hours).

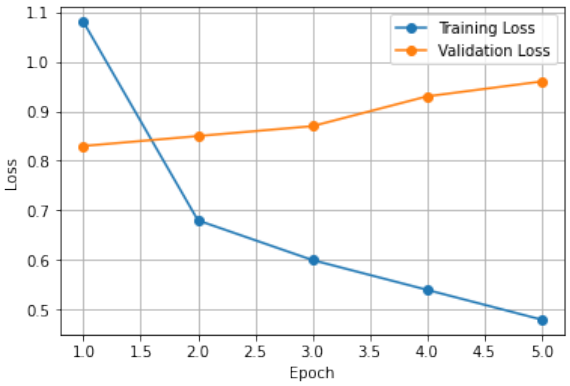
As our proposed approach has 10 times increasing speed, it is trained for 15 epochs, amounts to 8 hours. Our experiments were carried out in a Tesla T4 GPU with 16GB RAM.

4.2 Evaluation Metrics

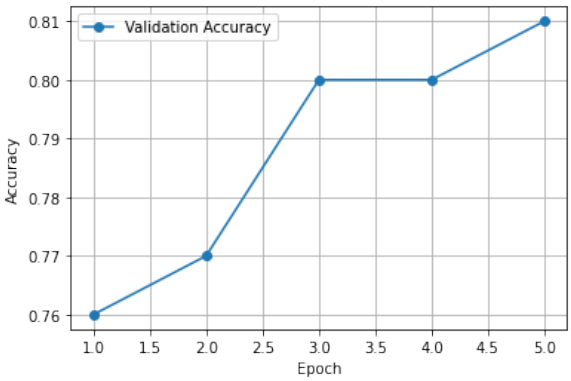
Given the task being multi-class classification, we used the overall classification accuracy of the test dataset on 11 topics as our main metric. We also calculated precision, recall, and F1 score for each of the 11 classes to examine if there exists any difference in performance.

4.3 Baseline Results

Our baseline model uses Longformer to compute local and global attention and generate feature representation, followed by a fully-connected layer to predict the probability for each of the 11 classes. We used two settings to train the model, one with the pre-trained weights and one without, and com-



a Training and validation loss



b Validation accuracy

Figure 3: Results for baseline model trained with Longformer architecture with pre-trained weights

pared the performance and efficiency.

4.3.1 With Pre-trained Weights

First, we initialized our model using the pre-trained `longformer-base-4096` weights. The linear layer is initialized with the default `kaiming_uniform` distribution. In order to fit in the Colab GPU and high RAM runtime, we set the sequence length to 4096 with a window size of 128. We followed the setting for the summarization task in the original Longformer paper (Beltagy et al., 2020), which set the start token `<s>` in each training sample to have global attention and the rest of the tokens in each window to have local attention.

We chose the optimizer to be `AdamW` and the loss function to be `CrossEntropyLoss` to train the baseline model. An initial learning rate of 5e-6 and 1e-4 is applied to update the weights of the Longformer layers and linear layer respectively. A learning rate scheduler with exponential decay is also applied with the decay rate γ of 0.8. The model is trained for 5 epochs, where after each epoch the model is evaluated using the validation Dataset, and the model with the highest validation accuracy is saved.

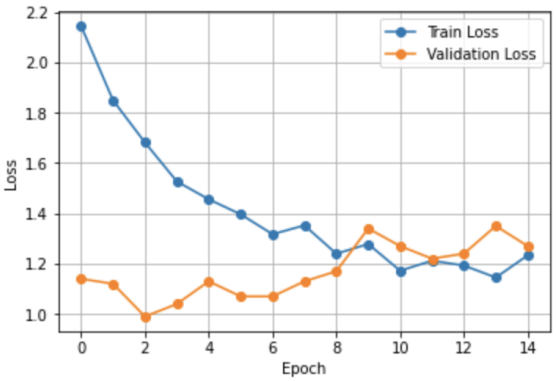
Figure 3 shows the plots of the loss and accuracy for the baseline model using pre-trained weights during the 5 epochs of training. In Figure 3a, we can see that the training loss continuously decreased whereas the validation loss slightly increased, showing a small level of over-fitting. Figure 3b shows that the validation accuracy improved and quickly converge at around 80%. The overall and class-specific precision, recall, and F1 score is also calculated and shown in Table 1. The baseline model with pre-trained weights achieved an overall 80.91% precision, 80.91% recall, and 81.99% F1 score, whereas the random guess on the 11-class classification task would be 9.0%. For class-specific performance, the baseline model shows some level of unbalance. Specifically, the baseline model performs better on math-related classes and performs relatively poorly on classifying `cs.AI` and `cs.NE` class. This may partially be due to the fact that there are more cs-related classes, and consequently, the differences between these classes are more subtle, making it harder for the baseline model to distinguish.

Class	Size	Precision	Recall	F1
cs.AI	100	66.67%	48.00%	55.81%
cs.CE	100	71.96%	77.00%	74.40%
cs.DS	100	81.91%	77.00%	79.38%
cs.IT	100	76.92%	80.00%	78.43%
cs.NE	100	67.01%	65.00%	65.99%
cs.PL	100	85.84%	97.00%	91.08%
cs.SY	100	89.01%	81.00%	84.82%
cs.cv	100	76.23%	93.00%	83.78%
math.AC	100	93.88%	92.00%	92.93%
math.GR	100	90.57%	96.00%	93.20%
math.ST	100	87.50%	84.00%	85.71%
overall	**1100**	**80.91%**	**80.91%**	**81.99%**

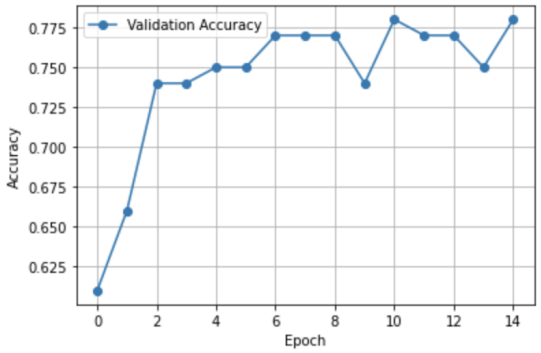
Table 1: Baseline model performance on test dataset with pretraining

4.3.2 Without Pre-trained Weights

To adjust for computing resources and training epochs, we also trained a baseline Longformer model from scratch without the pre-trained



a Training and validation loss



b Validation accuracy

Figure 4: Results for HSW model trained from scratch

weights. It provides a fair comparison with our model under the same computing resources on the same dataset. We used the same settings in sequence length, window size, and learning rates. We trained the model for 10 epochs instead of 5 for the model with pre-trained weights in order to acheive a better performance.

Class	Size	Precision	Recall	F1
cs.AI	100	33.96%	36.00%	34.95%
cs.CE	100	63.64%	42.00%	50.60%
cs.DS	100	75.00%	63.00%	68.48%
cs.IT	100	95.65%	66.00%	78.11%
cs.NE	100	44.44%	56.00%	49.56%
cs.PL	100	63.16%	96.00%	76.19%
cs.SY	100	74.04%	77.00%	75.49%
cs.cv	100	73.21%	82.00%	77.36%
math.AC	100	95.29%	81.00%	87.57%
math.GR	100	86.36%	95.00%	90.48%
math.ST	100	90.70%	78.00%	83.87%
overall	**1100**	**70.18%**	**70.18%**	**74.24%**

Table 2: Baseline model performance on test dataset without pretraining

The performance is shown in Table 2. The baseline model without pre-trained weights achieved an overall 70.18% precision, 70.18% recall, and 74.24% F1 score, which is lower than the baseline model with pre-trained weights. When looking into each categories, this model performs worse on almost all categories except `math.AC`. This is expected as the computing resource used in training this model is limited, and much less than the one used to train the `longformer-base-4096` weights in the original Longformer paper (Beltagy et al., 2020). Fewer training time and smaller training dataset result in a degradation in model performance.

The comparison between the baseline model with and without pre-trained weights suggests how much the limitation of our computing resources affects the model performance. Based on this difference, we can expect a better result of our HSW model described in the next section with better computing resources.

4.4 HSW Results

The model architecture of our Hierarchical Shifted Window (HSW) model follows the structure discussed in detail in section 3. For the experimental settings of HSW model, we randomly selected 2300 samples for training, 100 samples for validation, and 100 samples for testing. The window size was set to 64, the batch size was set to 32, and the sequence length of the input was set to 4096. We chose the optimizer to be `AdamW`, the loss function to be `CrossEntropyLoss` and the learning rate to be 3e-6. Cosine Annealing was adopted as the learning rate scheduler with the decay rate γ of 0.8. The model was trained for 15 epochs, with the model's performance on the validation dataset being evaluated after each epoch. The model with the highest validation accuracy was saved at the end of training.

Figure 4 contains the loss and accuracy plots for HSW model over 15 epochs. Figure 4a shows that the training loss decreases over time but the validation loss slightly increases, indicating a small

over-fitting. Figure 4b shows the increasing validation accuracy, indicating a strong performance. The metrics table for the HSW model, which includes the overall accuracy and F1 score as well as the performance on each of the 11 classes, is shown in Table 3. The overall accuracy of HSW model is 83.45%, and the F1 score is 84.86%. HSW model outperforms the baseline models in terms of both accuracy and F1 score, indicating that it is able to learn more complex patterns in the data and make more accurate predictions. For most of the classes, HSW model shows significantly better results than the baseline models in terms of precision, recall, and F1 score. Overall, the results show that HSW model is an effective and efficient model for long-sequence learning tasks.

Class	Size	Precision	Recall	F1
cs.AI	100	62.65%	52.00%	56.83%
cs.CE	100	82.98%	78.00%	80.41%
cs.DS	100	91.84%	90.00%	90.91%
cs.IT	100	88.17%	82.00%	84.97%
cs.NE	100	66.97%	73.00%	69.86%
cs.PL	100	83.93%	94.00%	88.68%
cs.SY	100	86.52%	77.00%	81.48%
cs.cv	100	71.79%	84.00%	77.42%
math.AC	100	97.94%	95.00%	96.45%
math.GR	100	95.19%	99.00%	97.06%
math.ST	100	90.38%	94.00%	92.16%
overall	**1100**	**83.45%**	**83.45%**	**84.86%**

Table 3: HSW model performance on test dataset

Table 4 shows the inference time and training time with batch size set to be 2 for our HSW model as well as the two baseline models. For training time, it was measured for each step. As shown in table 4, the two baseline models perform similarly for both time measures. HSW model is 25 times faster than the baseline models for inference time, and 30 times faster for training time per step. Such significant improvements again show an outstandingly low computational complexity for our model.

5 Discussions

Our HSW model shows better results than Longformer, the previous SOTA model on long-sequence text, when training on the Arvix Dataset on the multi-class classification task. While the results we obtained probably do not reflect these models' full potential due to our limited computation resources and relatively small training epoch, they provide strong evidence that the shifted window attention mechanism, originally designed for CV tasks (Liu et al., 2021), can also be a good solution to solve the high computation complexity issue in long sequence text.

	Inference Time	Training Time
Baseline without Pre-training	0.5487	1.9804
Baseline with Pre-training	0.5953	2.0201
HSW Model	0.0237	0.0664

Table 4: Comparison among models for inference time and training time

We think the advantage of the HSW model mainly lies in two folds. First, it further reduced the training time due to the smaller model size. The hierarchical structure of our HSW model not only allows it to learn representation at different semantic levels such as paragraph and article, it also decreased the number of model parameters that need to be learned during training and consequently reduces the amount of training time. In our experiment, our HSW model only requires 0.2 seconds per step, whereas both the pre-trained and not pre-trained Longformer model requires 2 seconds per step when trained on the same GPU device. This yields a 10-time improvement in training speed, which can save a significant amount of computation resources and can be especially beneficial for models designed for problems that involve long-sequence text.

Second, the success of Shifted Window model in both Computer Vision and Natural Language Processing tasks provides the possibility of a unified modeling approach. Swin Transformer was able to achieve SOTA performance in a number of Computer Vision problems, and our experiment also demonstrate its potential to achieve good performance in language-related tasks. Therefore, it would be advantageous if the Shifted Window model can be trained uniformly to represent both text and image features, which can easily allow weights sharing across these different modalities.

6 Impact Statement

One major limitation of our project comes from the time and computation restrictions. Our models are trained with relatively small training epochs, and we believe that it is possible to further improve the model performance with more hyperparameter tuning and training epochs. Therefore, while the Sifted Window model shows faster improvement and better classification performance in our results, we cannot eliminate the possibility that Longformer can exceed our HSW model with more training.

Also, we only tested our HSW model on multi-class classification tasks in this project. We did not verify whether its hierarchical structure and representation can perform well on other generative tasks such as summarization and translation. This is important if we want to use Shifted Window model as a unified modeling approach to learn both image and text and apply it in tasks such as image captioning.

References

Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, Kaiser Lukasz, and Illia Polosukhin. 2017. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Jun He, Liqun Wang, Liu Liu, Jiao Feng, and Hao Wu. 2019. Long document classification from local word glimpses via recurrent attention learning. *IEEE Access*, 7:40707–40718.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022.

Sandeep Subramanian, Ronan Collobert, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2020. Multi-scale transformer language models. *arXiv preprint arXiv:2005.00581*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.