Probing Position-Aware Attention Mechanism in Long Document Understanding

Anonymous ACL submission

Abstract

Long document understanding is a challenging problem in natural language understanding. Most current transformer-based models only employ textual information for attention calculation due to high computation limit. To address those issues for long document understanding, we explore new approaches using different position-aware attention masks and investigate their performance on different benchmarks. Experimental results show that our models have the advantages on long document understanding based on various evaluation metrics. Furthermore, our approach makes changes only to the attention module in the transformer and thus can be flexibly detached and plugged into any other transformer-based solutions with ease.

1 Introduction

Despite the rapid advancements in deep learning, particularly in the fields of computer vision and natural language processing, the tasks of understanding multimodal data, e.g. vision-language tasks incorporating scene text or multimodal nature language understanding, have not been fully developed with the same pace. It is due to several inherent challenges (Baltrušaitis et al., 2018). First, efficient feature encoding and extraction for each modality is difficult. Second, data always contain both text and visual information and the alignment between them is not easy, for example between text and audio or video. And the final challenge is the intricate obstacle of how to fuse all modalities of different formats for the tasks' objective.

Sharing the same challenges with multimodal data, however, recent years have witnessed many developments in the new task of document understanding (or in some contexts known as document intelligence). Most of them are largely benefited from the fast development of transformer architectures (Tay et al., 2020). Nonetheless, transformer suffers from quadratic computation of attention mechanism, the heart of its architecture, in the face of its well-proven power across diverse tasks.

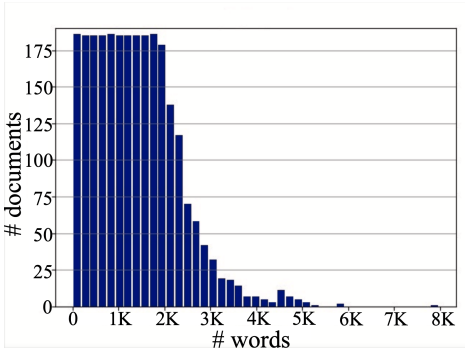


Figure 1: Distribution of document lengths in RVL-CDIP dataset (Harley et al.), a subset of IIT-CDIP used predominantly in the document understanding pretrain task–and in all of our models. While most current models limit the document lengths to maximum 512 tokens because of memory and computational issues, most of documents in this datasets have a much longer lengths, posing the necessity for a model capable of processing longer documents.

Apart from computation issues, dealing with long documents in the context of document understanding requires a linking mechanism between information across pages/segments in terms of both text and spatial information. Given the lengths of long documents as illustrated in Figure 1, it is reasonable to assume that useful information is spanned across their lengths. Likewise, most current document understanding models are short models, such as LayoutLM (Xu et al., 2020b,a), which limit the document length to only 512 words and thus discard lots of useful information.

In natural language processing, there have been some recent work to reduce this cost by sparsifying the attention matrix with predefined sliding windows, such as Longformer (Beltagy et al., 2020), BigBird (Zaheer et al., 2020) or ETC (Ainslie et al., 2020). These models push the limit of doc-

ument length from 512 to 4096 words with optimized memory and computation costs. Furthermore, some other recent attempts, e.g. in Nguyen et al. (2021), have not been successful in processing long documents that are longer than 2048, partly because they add another small transformer module, which consumes many resources on top of the current transformer infrastructure.

In this paper, we explore new approaches using different position-aware attention masks and investigate their performances on long document understanding. Unlike those attempts to introduce new pretraining tasks to enforce better representation learning, or new complicated infrastructures into the already-heavy transformer architectures, e.g. (Appalaraju et al., 2021), our motivation is to firstly retain most of the simplistic architecture of LayoutLM (Xu et al., 2020b) and then flexibly bring textual and/or spatial information into attention module in efficient ways that do not affect any other module. By doing so, we could enhance the practicality of transformer-based models while achieving the needed power of handling long documents.

In summary, our contributions are as follows. 1) We motivate the new use of spatial information into transformer's attention in a simplistic way, making it as a plug-able module to any transformer architectures. 2) We are able to tackle the document understanding task with input data having up to 4096 words. 3) Experimental results prove the advantages of our approaches on various long-document datasets in comparison to short models. And finally, our implementation and pretrained models will be open to public [1].

2 Related Work

Transformer Architecture In 2017, Transformer was first proposed to replace Seq2Seq model (Sutskever et al., 2014) in natural language processing (NLP) (Vaswani et al., 2017). Since then, transformer has increasingly become a vital part of deep learning solutions (Tay et al., 2020) in NLP (Beltagy et al., 2020; Tay et al., 2020; Kitaev et al., 2020; Zaheer et al., 2020; Zhang et al., 2021), computer vision (Parmar et al., 2018; Tay et al., 2020; Katharopoulos et al., 2020), speech processing (Katharopoulos et al., 2020), and genomics (Zaheer et al., 2020). Recently, many new work on transformer focus on addressing its main drawback of high computational expense, in which the attention operation takes quadratic time $\mathcal{O}(n^2)$, where n is the sequence length.

For example, Longformer (Beltagy et al., 2020) uses sliding-window or dilated sliding window to capture only small context of each word and reserves only some sparse global connections. Having similar high-level idea, ETC (Ainslie et al., 2020) also embeds relative position to input sequences and adds contrastive predictive encoding. With the same key idea of limiting the global, fully-connected attentions, BigBird (Zaheer et al., 2020) optimizes sliding window mode with random connections to make it sparser while not harming the performance. Likewise, less global and more local attentions are learned for higher dimensions to achieve good results (Parmar et al., 2018). Another orthogonal approach is to approximate attention kernels such as substituting softmax with low-rank kernels (Katharopoulos et al., 2020) or extracting random, orthogonal features (Choromanski et al., 2020). Different from the aforementioned methods, our approach not only reduces the memory consumption by narrowing the attention's context window for each single token, but also exploits layout information flexibly and complements its benefits with the typical text information when needed.

Multimodal Document Pretraining Document understanding largely inherits from the development of multimodal pretraining, in which features of text and vision are learned and fused together for downstream tasks (Li et al., 2020; Chen et al., 2020; Luo et al., 2020). In 2020, LayoutLM (Xu et al., 2020b) was proposed, which is the first work to pre-train document layouts along with other features. Later, LayoutLMv2 (Xu et al., 2020a) motivated the use of spatial information into attention by introducing learnable relative bias terms, which have limited power due to the small number of parameters added and the shallow level of attention intervention. On the opposite, our work brings spatial information to the granular attention level where the main learnable parameters, namely query, key and value matrices can all be altered.

Recently, Docformer (Appalaraju et al., 2021) and StrucuturalLM (Li et al., 2021) were proposed to address the document learning problem with a two-pronged approach: introducing new pretrain tasks as well as make suitable changes to the processing or embedding layers. By introducing more

[1] Our code is submitted along with this paper in the supplemental material and will be made publicly available.

processing layers and do not tackle attention parts, however, they still suffer from high computational cost of the attention, the heart of transformer. Probably Skim-Attention (Nguyen et al., 2021) has the most related motivation to ours, although we have a more memory-efficient and faster way of handling layout information directly from input and not from after the embedding like theirs.

3 Our Model

In this section, we introduce new components into the pretrain model to enable it with the capacity of dealing with long documents effectively and subsequently describe those components in the following subsections.

3.1 Pretrain Model Architecture

Inspired by LayoutLMv1 model (Xu et al., 2020b), we use the similar Masked Language Modeling (MLM) pretrain task, in which 80% tokens are masked with the same `[MASK]` token, 10% tokens are masked randomly and the rest 10% is unchanged. For embedding layer, we use the same word embedding, word 1D position embedding based on words' positions in the documents and the 2D position embedding based on word-bounding box alignment.

Different from LayoutLMv1 model, our model has two targeted designs to deal with long documents. First, we extend the model capability of dealing with documents from maximum 512 tokens to 4096 and beyond. We choose the sliding-window method, inspired from Beltagy et al. (2020), given its lightweight and elegance in limiting the context window many times, making it significantly more memory friendly without sacrificing the performances. The second design is introducing new spatial-based distance masks in our model. These new distance masks are different from sliding-window-based ones, in which the context is strictly based on neighboring words and therefore cannot be changed. The following section 3.2 will elaborate the establishment and usage of these new distance masks and comparison with other masks.

It is also worth mentioning that the post-OCR processing is also important for long documents. To our best knowledge, all available OCR engines only produce the bounding box coordinates on page-level basis, meaning that there is no connection among the pages in the documents in the OCR-generated results. For short documents, it is reasonable to assume that each page typically contains 512 words or less, and hence there is no need to alter the bounding boxes (except for normalization). On the contrary in long documents, it is crucially important to take into account the page sizes and indices of all pages per document to adjust the bounding boxes accordingly.

3.2 Different Attention Masks

We observe that the relationship of words follows not only in the consecutive nature of texts, but also in the sections organized in the layout organization. In addition, document understanding largely replies on the quality of OCR engines. To compensate the inaccuracies of OCR results, spatial layout information is an important complementary information to the normal textual information. As shown in Figure 2, spatial information based on bounding boxes' coordinates offer a different angle in terms of relationship, in that many neighbors are different and beyond the coverage of the fixed-width sliding window mechanism.

Original Attention Masks In the Transformer (Vaswani et al., 2017)-based architecture, in each single layer, the attention score is calculated by two main steps, as in Equations (1) and (2):

$$\text{score}(\mathbf{Q}, \mathbf{K}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right) \quad (1)$$

$$\text{attn\_score}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{score}(\mathbf{Q}, \mathbf{K}) \cdot \mathbf{V}, \quad (2)$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ stand for the learnable Query, Key, and Value matrices respectively. Given the lengths of these three matrices are all N, which is also the input length, the complexity of each step is $\mathcal{O}(N^2)$. In practice, each transformer model has many layers. Each layer has a number of heads to increase the model's learning capacity with more parameters. Nonetheless, all of these factors do not change the given complexity of the attention computation–the major cost of the whole transformer architecture.

Sliding-Window Masks (Figure 2a) We usually call the aforementioned original attention mechanism as full attention since each input token attends to all N available tokens including itself. In the sliding-window approach as inspired from Beltagy et al. (2020), we limit the context for each token from the full length of N to M including $M/2$ tokens before and $M/2$ after it (not including itself) with the assumption that reasonable short context is sufficient comparable with the full attention while reducing the computational cost. By

doing so, the complexity of Equation (1) and (2) is reduced to $\mathcal{O}\left(N(M+1)\right)$, which significantly decreases memory and computation cost, especially when $M << N$ as we usually choose in practice, e.g. $N = 4096, M = 512$.

$$\mathbf{K}_w = \text{extract\_window}(\mathbf{K}) \tag{3}$$

$$\text{score}(\mathbf{Q}, \mathbf{K}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}_w^T}{\sqrt{d_k}}\right) \tag{4}$$

$$\mathbf{V}_w = \text{extract\_window}(\mathbf{V}) \tag{5}$$

$$\text{attn\_score}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{score}(\mathbf{Q}, \mathbf{K}) \cdot \mathbf{V}_w \tag{6}$$

Using that intuition, the calculations are now changed to Equations (3–6), with the added `extract_window` steps in Equations (3) and (5). The `extract_window` step essentially reduces the context for each token from N down to the window size M, leading to the complexity of Equations (4) and (6) reduced to $\mathcal{O}(N(M+1))$ each[2].

Sliding-Window plus Random Token Masks (Figure 2b) Inspired from Zaheer et al. (2020), on top of sliding windows, we add some more random tokens with the hope that they will enhance the power of attention.

$$\mathbf{K}_w = \text{extract\_blocked\_rand\_window}(\mathbf{K}) \tag{7}$$

$$\mathbf{V}_w = \text{extract\_blocked\_rand\_window}(\mathbf{V}) \tag{8}$$

We make a change to the aforementioned `extract_window` procedure. We divide the original sequence length to blocks (e.g. 512 to 8 equal blocks of length 64), to facilitate grouping and chunking, as well as to lessen the computational steps (have much less sliding windows). As a result, Equations (3) and (4) are now replaced by (7) and (8). What is more, in these new procedures, some random blocks are also marked on top of sliding blocked windows.

Spatial-based Distance Masks (Figure 2c) For each document in our model, we generate a distance-based attention mask, which shares the same shape with sliding window (if having the same number of neighbors). Here, we call the tokens within the context for each token as its neighbors. This process comprises of a couple of steps. First, we identify the center point of all bounding boxes. Second, we fit the kNN algorithm to the sequence of those points based on the L2 distance, which correspond to the sequence of the aligned words, resulting in a 2D distance matrix. Finally, we record the neighbor indices for each node and end up having a $N \times M$ matrix, where N is the number of tokens and M is the number of neighbors.

$$\mathbf{K}_w = \text{extract\_neighbors}(\mathbf{K}) \tag{9}$$

$$\mathbf{V}_w = \text{extract\_neighbors}(\mathbf{V}) \tag{10}$$

These steps are summarized in the same function `extract_neighbors`, as shown in Equations (9) and (10), which replace the `extract_window` in Equations (3) and (5). The rest steps stay the same. Intuitively, equations (9) and (10) return the identically shaped matrices as equations (3) and (5). But they use kNN-based spatial contexts based on layout information instead of textual contexts.

Implementation of Distance Masks In terms of implementation, there are certain considerations to enable the use of distance masks, which consumes more memory compared to the normal sliding window mechanism as detailed below.

First, identifying spatial neighbors for each token usually takes quadratic time, which is a great deterrent to our solution. So we choose to use `scikit-learn`'s kNN library [3] for its well-regarded efficiency and speed.

Second, "where to create distance masks: in dataset loader or in model computation" is a key problem. We choose to create distance masks in dataset loader for the following reasons. First, the main obstacle of applying long-document attention methods to LayoutLM or other document understanding models is that these models are inherently heavy in terms of memory consumption. If place the quadratic computation for distance mask in the model phase, the model will be significantly slower (in proportionate to the document limit length) and the risk of out-of-memory will be much higher (given the limitation of GPU memory nowadays). Second, by preemptively computing the distance mask in the dataset loader, e.g. using Pytorch Dataloader [4] and exploiting its data buffering mechanism, the data loading will not be slower by running multiple loader processes simultaneously.

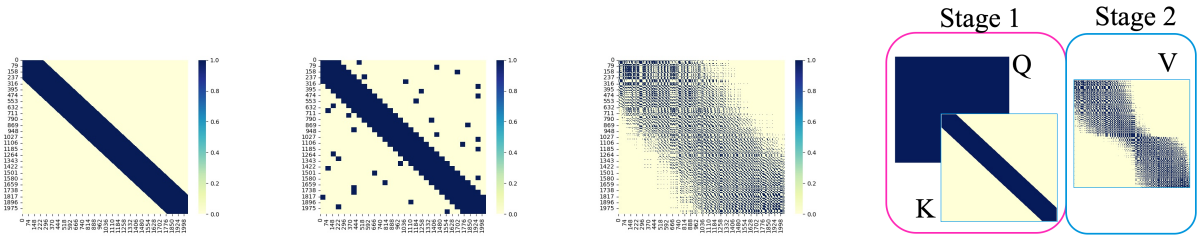
Finally, for the sliding-window attention mask, we inherit a well-regarded implementation from Huggingface [5], considering our base model LayoutLMv1 model is based on this framework. Then

[2] To enable fast calculations in Equations (4) and (6) with now-changed matrix shapes, one has to extract and chunk the contexts for all tokens in a way that can exploit fast matrix multiplication (e.g. by using `einsum` in `pytorch`).

[3] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[4] https://pytorch.org/docs/stable/_modules/torch/utils/data/dataloader.html#DataLoader

[5] https://huggingface.co/transformers/model_doc/longformer.html



(a) SW (b) SW+RAND (c) DISTANCE (d) DISTANCE+SW

Figure 2: Visualization of our models' different types of attention mask for real samples from RVL-CDIP dataset (Harley et al.) with limit length of 2048 and context size 512 (for both textual and spatial cases). Fig 2a is sliding window (SW), Fig 2b is sliding window in blocks with 1-per-block random blocks(SW+RAND), Fig 2c is spatial-based distance mask, and Fig 2d is the combination of sliding window and distance modes (applied to stage 1 and stage 2 of attention respectively). *Legend*: Attention mask may only have values of 0 and 1, which are represented as the light-yellow background and dark-blue foreground colors, respectively.

we implement our distance-based solution on top of it.

3.3 Pretrain Model Variants

We probe the above four position-aware attention mechanisms in our new long-document models, as demonstrated in Figure 2. These attention variants are used directly for the MLM pretraining tasks, and subsequently used as the backbone for downstream tasks (see Section 4). As explained in Section 3.2, the changes are only made to the attention module, thus the rest of the MLM architecture stays the same as in Xu et al. (2020b). Therefore our approach can be used as an off-the-shelf solution to any transformer architecture, whenever layout information is available.

SW Model. This model directly uses sliding window (SW) mode for attention masks, which significantly reduces the computation and is shown to be effective for long documents. Detailed computation is illustrated in Equations (3–6).

SW+RAND Model. This model uses blocked sliding windows and some random blocks on top[6]. These changes to sliding window appear in Equations (7) and (8).

DISTANCE Model. In this model, we completely replace the normal sliding-window attention mask with our distance-base mask as elaborated in Equations (3–6), with the notable change that the spatial neighbors are preemptively computed using the kNN algorithm in the dataset processing phase.

DISTANCE+SW Model. In this model, we combine the distance-based and textual attention masks together in the attention operations. In detail, in the sequential steps in Equations (3–6), we replace Equation (3) with Equation (9), making the first attention step use the distance-based mask while retaining the second step. These two steps are independent of each other and both preserve the logic and matrices' shapes of the attention module. Our intuition is to combine both spatial and textual masks in one single attention pass.

4 Experiments

4.1 Tasks and Datasets

We present the tasks and the associated datasets used in our experiments, starting from IIT-CDIP for the model pretraining to the other datasets used in different downstream tasks and ablation studies.

Pretraining We use **IIT-CDIP Test Collection 1.0** [7] dataset for our MLM pretraining task. This is a large scale dataset that has over 6 million multi-page documents and around 11 million pages in total (each page is stored as an image). We use the same OCR engine used in LayoutLM model (Xu et al., 2020b) to extract the bounding boxes coordinates and other metadata such as page sizes, lines, etc.

For the MLM task, we retain most of the settings of the LayoutLM model with 12-layer Transformer with 12 attention heads and hidden sizes of 768. In terms of spatial embedding for bounding boxes, we scale the bounding boxes for all pages to $[0, 1023]$. We pretrain the model with four attention mechanisms described in Section 3 using 8 parallel Tesla V100 GPUs with a combined batch size of 64 and learning rate 5e-5.

Document Classification For this task, we use **RVL-CDIP** (Harley et al.) dataset, which is a subset of the pretraining dataset IIT-CDIP and is de-

[6] We use 3 random blocks in our implementation before splitting and chunking the blocked chunks

[7] https://ir.nist.gov/cdip/

signed for the document classification task. It comprises of 16 classes and each class equally has 25K grayscale images. These total 400K images are split into 320K images for training and 40K images each for validation and testing. The document length distribution of this dataset is shown in Figure 1.

For this fine-tuning classification model, we concatenate the output from the pretrained model before applying the typical softmax-crossentropy layer. We only finetune for 30 epochs with the learning rate 1e-5 and batch size of 32.

Sequence Labeling There are two different datasets that we use for this task, namely Kleister-NDA and FunSD.

Kleister-NDA (Graliński et al., 2020; Stanisławek et al., 2021)[8] This dataset is presented with a task aiming to extract long documents that contain layout information, which is closely related to our models. The task associated with this dataset is to extract values for four given classes. Kleister-NDA has 540 documents in total (254 training, 83 validation, and 203 testing) with 2,160 entities annotated and average of 2,540 words per document. Unfortunately, the evaluation of this dataset provided by the authors requires post-processing, which is not published yet. We have to cast this task as an entity-labeling task in FunSD below and evaluate it with F1 score for reproducibility purpose. As a result, for this dataset we report all reproduced results using the same preprocessing and metric calculation. Finally, we employ the same OCR engine used in IIT-CDIP and RVL-CDIP datasets.

FunSD (Guillaume Jaume, 2019)[9] This is a lightweight dataset that consists of noisy scanned documents and is designed for form understanding task with 7 different classes. It has 199 scanned forms that contain more than 31K words and 9.7K entities and 5.3K relations in combination. Although FunSD can hardly be considered as a long-document dataset (each single document only has less than 512 words), it is still useful for our ablation studies to compare state-of-the-art models that employed this dataset (see Section 4.4).

4.2 Baselines

As usual, we use downstream tasks to evaluate and compare our 4 model variants (see Figure 2) with the following baselines:

[8] https://github.com/applicaai/kleister-nda

[9] https://guillaumejaume.github.io/FUNSD

Type	Model	SeqLen	Acc (%) ↑
Text	BERT-base	512	89.81
	RoBERTa-base	512	90.06
	BERT-large	512	89.92
	RoBERTa-large	512	90.11
	Bigbird-base	4096	93.48
	Longformer-base	4096	93.85
	Bigbird-large	4096	93.34
	Longformer-large	4096	93.73
Text+Layout	LayoutLM-base	512	91.88
	LayoutLM-large	512	91.90
	Ours SW	4096	94.50
	Ours SW+RAND	4096	**95.25**
	Ours DISTANCE	4096	94.79
	Ours DISTANCE+SW	4096	94.69

Table 1: Classification accuracy for RVL-CDIP. For this long-document dataset, the models capable of using 4096 words uniformly beat other models and layout information helps with the task compared with using Text input. All our long models show their advantages on this long dataset.

Text: This group consists of the traditional models which only accept text input with maximum length of 512 including BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), and other long models including Bigbird (Zaheer et al., 2020) and Longformer (Beltagy et al., 2020), which can support longer documents (we pretrain them up to the length 4096).

Text+Layout: This group contains models that accept both text and layout information, including LayoutLM (Xu et al., 2020b) that accepts only 512 words per document.

An important note is that our target is to tackle long documents, in which computational cost of processing and memory is the main obstacle, even for the "base" versions (typically the "large" counterparts would have at least about 2X-3X number of parameters, and hence much more memory intensive). To make fair comparisons and make our models more practical, we only compare with the large models (i.e. the ones capable of dealing documents with length 4096) using only "base" versions.

4.3 Results and Discussions

Document Classification As shown in Table 1, for this long-document dataset, long models (ones that accept long input to 4096 tokens) clearly outperform short models in both baseline groups and our models, with or without layout information added to the input.

From these results, We find two meaningful observations. First, long documents have useful information spanned across document. Second, our

Type	Model	SeqLen	F1 ↑
Text	BERT-base	512	47.06
	BERT-large	512	52.66
	Longformer-base	4096	61.78
	Bigbird-base	4096	46.98
Text+Layout	LayoutLM-base	512	55.69
	LayoutLM-large	512	61.95
	Ours SW	4096	**64.06**
	Ours SW+RAND	4096	58.92
	Ours DISTANCE	4096	57.01
	Ours DISTANCE+SW	4096	44.70

Table 2: Results on Kleister-NDA dataset (validation split) with entity-labeling performance. Although this dataset is very challenging, the long models still show the advantages over the short ones.

Type	Model	SeqLen	F1 ↑
Text	BERT-base	512	60.3
	RoBERTa-base	512	66.5
	BERT-large	512	65.6
	RoBERTa-large	512	70.7
	Bigbird-base	4096	45.8
	Longformer-base	4096	71.4
	Bigbird-large	4096	46.8
	Longformer-large	4096	73.5
Text+Layout	LayoutLM-base	512	78.7
	LayoutLM-large	512	**79.0**
	Ours SW	4096	69.9
	Ours SW+RAND	4096	77.1
	Ours DISTANCE	4096	64.0
	Ours DISTANCE+SW	4096	61.8

Table 3: Comparison on FunSD dataset. As usual, layout information is helpful in boosting the performance given text. However, long models do not perform well compared with short models on this short-document dataset.

models show advantages over others with the capability of absorbing and processing long documents. Therefore, our models are more practical in dealing with real-world data, which usually have more text than only 512 words.

Sequence Labeling The results are shown in Table 2, which is based on the validation split since there is no annotation for the test split. Compared with other reported results (e.g in (Xu et al., 2020a; Appalaraju et al., 2021)), our reproduced results are much lower, posing this task is a challenging one (it has decoyed texts that have no associated labels). Comparing the "base" versions, our models still clearly show advantages over the baselines. In particular, our SW model achieves the highest scores by employing 4096 tokens as well as combining text and layout information.

Furthermore, we find that our DISTANCE+SW are outperformed by other models. One possible reason is that the OCR engine couldn't understand the decoying annotation. Consequently, it generates the normal OCR results that do not correlate with the text.

4.4 Ablation: Long Models on Short Dataset

Table 3 shows that on FunSD (a small and short document dataset), long models do not perform well compared to short models. However, we still observe that layout information generally helps with the task. The main reason is that long models essentially have much more parameters. 1000-step fine-tuning with only 199 samples can hardly tune parameters well.

Especially, when long models are forced to have most input as padding (e.g. 512 words + 3584 pad tokens), it makes fine-tuning more difficult. Even if we reduce the maximum input length to 512, the final results for long models do not change much. One of the main reasons is that long models have their embedding trained for 4096 tokens, and hence making it to be fine-tuned for only 512 tokens for a few steps generally does not work well.

In the next ablation studies, we explore the implications of the newly added spatial information into attention in our models as compared with some baselines.

4.5 Ablation: Different-Length Documents

In this study, we keep all documents intact and categorize them based on their original lengths. The purpose is to explore how the models work if we do not cut any information from document based on the capacity limit of the models and take all possible available information in each document. This analysis offers a different perspective from the setting in Section 4.6, where we do not respect the input and have to purge the excesses to make them compatible with the model. As a result, for 40K test samples in RVL-CDIP dataset, we only have 9268 samples having lengths ≥ 512, and 2312 samples with lengths ≥ 1024, and only 106 samples with lengths ≥ 2048. Note that by this grouping method, the latter group is a subset of the immediate former one, e.g. the group with 106 samples is a subset of the group with 2312 samples.

As shown in Figure 3, the consistent observation is that our models perform worse when the original document length increases, although the differences are not significant. There could be several possible reasons for this behavior: the models are not well pretrained and/or fine-tuned; the

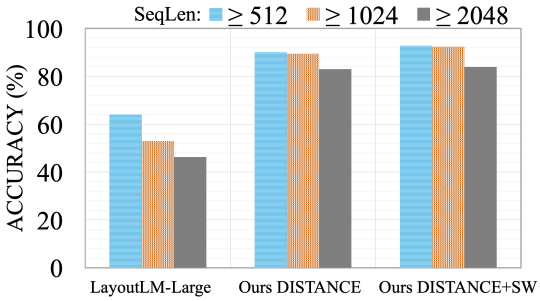


Figure 3: RVL-CDIP performance on different documents types based on their original lengths (i.e. without purging) using LayoutLM (with the best "large" version) and our models (DISTANCE and DISTANCE+SW). Our models clearly outperform LayoutLM models in all categories.

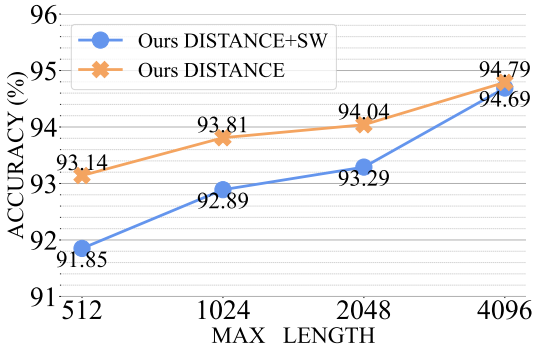


Figure 4: RVL-CDIP performances on different maximum lengths using our DISTANCE and DISTANCE+SW models. For each case of lengths 512, 1024, 2048 and 4096, the test set contains the same 40K samples. Longer maximum length gives better results.

long documents have lots of contents and many of which confuse the classifier; or there are many noises from OCR results.

4.6 Ablation: Different Max Input Lengths

Given the pretrained models that can accept input up to 4096 tokens, we finetune them with input of different maximum lengths (`max_len`) . Likewise, if a document has the content longer than `max_len`, the extended part will be purged. As a result, in RVL-CDIP dataset, all 40K test samples stay the same for each test case.

As shown in Figure 4, our models show bigger power if we allow it to absorb more and more tokens from input. It matches with our intuition that for long documents, all parts contain useful information. We should not limit document lengths to 512 tokens, which unfortunately is a standard setting in many current document understanding models.

4.7 Further Discussion on Distance Masks

As seen in the above experimental results, direct usage of 2D layout context information into the transformer's attention has some advantages over the baselines. However, its performance does not match with the typical usage of 1D textual information. This might be discouraging at first since introducing spatial information brings two complications: heavy computation of 2D data preprocessing with kNN process and its obstacle to speed up. Consequently, for the accuracy-speed tradeoff, we limit the distance-based context to 128 (compared with 512 in textual contexts), which also makes the distance-based models suffer.

We hypothesize the drawbacks are due to some objective limitations. First, the performance of the whole pipeline heavily depends on the quality of the OCR pre-processing. In all datasets being used, texts/words are the ground-truth but the bounding boxes and their alignments with the text are made possible by those OCR engines. Second, we trade the accuracy of kNN sometimes for speed of processing 2D data. Finally, with a decoying design as in Kleister-NDA (Section 4.3), in particular, the OCR results are even less aligned with text. Consequently, we also conjecture that with further development in OCR technologies, the use of distance masks and layout information in general would be much more helpful in practice.

5 Conclusion and Discussion

We propose a versatile solution for document understanding task, in which the layout information can either replace or incorporate with the textual information for attention modules in a flexibly pluggable manner. Our solution has shown promising results on long document understanding tasks.

In our future work, we will further reduce memory consumption of these transformer-based models with heavy multimodal input. We will also spend effort to improve the speed of pretraining. Similar to LayoutLM (Xu et al., 2020b), pretraining of these models usually takes 80 hours to finish 1 epoch using 8 parallel V100 expensive GPUs, each of which has 32GB of memory. There are still a lot of room for improvement to make these models more efficient and practical.

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. Etc: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284.

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R Manmatha. 2021. Docformer: End-to-end transformer for document understanding. *arXiv preprint arXiv:2106.11539*.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2018. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443.

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

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Uniter: Universal image-text representation learning. In *European Conference on Computer Vision*, pages 104–120. Springer.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Filip Graliński, Tomasz Stanisławek, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartosz Topolski, and Przemysław Biecek. 2020. Kleister: A novel task for information extraction involving long documents with complex layout. *arXiv preprint arXiv:2003.02356*.

Jean-Philippe Thiran Guillaume Jaume, Hazim Kemal Ekenel. 2019. Funsd: A dataset for form understanding in noisy scanned documents. In *Accepted to ICDAR-OST*.

Adam W Harley, Alex Ufkes, and Konstantinos G Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In *International Conference on Document Analysis and Recognition (ICDAR)*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Chenliang Li, Bin Bi, Ming Yan, Wei Wang, Songfang Huang, Fei Huang, and Luo Si. 2021. Structurallm: Structural pre-training for form understanding. *arXiv preprint arXiv:2105.11210*.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Xilin Chen, and Ming Zhou. 2020. Univilm: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353*.

Laura Nguyen, Thomas Scialom, Jacopo Staiano, and Benjamin Piwowarski. 2021. Skim-attention: Learning to focus via document layout. *arXiv preprint arXiv:2109.01078*.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR.

Tomasz Stanisławek, Filip Graliński, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartosz Topolski, and Przemysław Biecek. 2021. Kleister: Key information extraction datasets involving long documents with complex layouts. *arXiv preprint arXiv:2105.05796*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. 2020a. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740*.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020b. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200.

Manzil Zaheer, Guru Guruganesh, 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. *arXiv preprint arXiv:2007.14062*.

Hang Zhang, Yeyun Gong, Yelong Shen, Weisheng Li, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2021. Poolingformer: Long document modeling with pooling attention. *arXiv preprint arXiv:2105.04371*.

Parameter Name	Value
do_lower_case	true
fp16	true
fp16_backend	amp
gradient_accumulation_steps	4
max_seq_length	4096
max_2d_position_embeddings	1024
max_steps	1000000
model_name_or_path	allenai/longformer-base-4096
dataloader_num_workers	64
tasks	mask_lm
optimizer	transformers_AdamW
learning_rate	5e-5
warmup_ratio	0.1
weight_decay	0.01
whole_word_masking	false
add_prefix_space	true
attention_window	512

Table 4: Main pretrain hyperpameters on the MLM pretraining task for the ITT-CDIP large-scale dataset. There are 3 variants share this set of parameters that are Ours SW, Ours DISTANCE and Ours DISTANCE+SW models. All of them use the pretrained weights from Longformer-base (Beltagy et al., 2020) model.

A More Information on the Pretrain Task

Pretrain Data Preprocessing As described, for pretrain model we retain the same OCR engine for generating and aligning layout and text information from LayoutLM (Xu et al., 2020b). The task is also the same, which is Masked Language Modeling (MLM). To deal with long documents, we have to implement the additional sliding-window, random-block and distance-based masks.

Pretrain Model Implementation Our solution only makes changes to the attention module, in which uses can choose to use any types of attention masks from the 4 variants illustrated in Figure 2.

For the SW and SW+RAND models which are also our new pretrain models, we implement the layout-related part on top of the original BigBird [10] and Longformer [11] implementations from Huggingface's transformers, respectively. Otherwise the distance-based masks, which are employed in DISTANCE and DISTANCE+SW models, are newly implemented as a pluggable module.

Training MLM We pre-train the task on the IIT-CDIP datasets, using a single-node multi-GPU mode. Each job was run on a server with 8 V100 Nvidia GPUs, each of which has 32GB memory and fast processors. For text-only models, please refer to LayoutLM's github [12].

For SW model, we use the public pretrained weights from Lomgformer (Beltagy et al., 2020). Other of our models employ the same set of parameters, except for the pretrained weights, in which SW+RAND model uses the weights from Bigbird (Zaheer et al., 2020) and the last two models having distance masks (DISTANCE and DISTANCE+SW models) use the same pretrained weights as SW model, as demonstrated in Table 4.

It is also worth noting that the pretrained weights from Longformer and Bigbird models are useful even for the models using distance masks because those two model families support documents with length 4096, so the position embeddings are helpful. For speed and memory tradeoff, we limit the context for distance masks to only 128 (vs. 512 in textual contexts), without sacrificing much performances, as reported in Section 4.3.

Training Notes Although not reported in the main content, we note some lessons learned from the pretraining task. As we observe, the Ours SW model consistently achieves the best results, while consuming the least GPU memory. For the base model, it only consumes about 7 GB GPU memory and Ours DISTANCE+SW that uses sliding-window attention on its half processing also consumes about 9 GB memory. Both models, as a result, can be deployed well on a broad range of GPUs in the market.

Unlike those conveniences, Ours SW+RAND and Ours DISTANCE do not share the same advantages. In fact, they consumes about more than 30GB GPU memory each, limiting their practicality. We hypothesize the main reason for such drawbacks is that they have random, inconsistent patterns, and hence there is no efficient way to take advantage of fast memory-efficient and fast matrix operations.

Finally, although showing promising practical behaviors, all baselines and our models, and almost any transformer-based ones are certainly not lightweight models. And although there are advancements in compressing those heavy models (e.g. (Touvron et al., 2021; Frankle and Carbin, 2018), there seems to be a considerable way to go for making these model run on mobile devices in the near future.

[10] https://huggingface.co/transformers/model_doc/bigbird.html

[11] https://huggingface.co/transformers/model_doc/longformer.html

[12] https://github.com/microsoft/unilm

B More information on Finetuning Tasks

As described in the main content, after pretraining, the saved models are the backbone for the respective fine-tuning model types. For that reason, the parameters are mostly shared with their pretrain counter-part models, e.g. Table 4 for Ours SW models. Generally, we keep the same optimizer and batch size of 32 (combined across all used parallel GPUs).

For **RVL-CDIP** in the document classification task, we use the `SequenceClassification` model type. On top of the pretrain skeleton, we add a small classifier with 2 fully-connected layers and a drop-out layer in between. The final output is the single class for the whole sequence/document.

For **FunSD** and **Kleister-NDA** datasets, we instead use the `TokenClassification` model type, which is designed to classify all-document entities. The similar classifier is added to the pretrained skeleton, now with a different usage in which each token/entity is to be classified into 1 of the number of given classes.

What's more, to preprocess these two datasets, we have to ingest all available document tokens. Likewise, with documents longer than the maximum lengths, we need to cut those documents, and recursively treat the overflowing parts in the same way. In terms of implementation, unlike FunSD that is lightweight, we always want to avoid loading the whole dataset into the memory but rather take advantage of the data buffering in feeding to the models. As a result, we pre-process all data first, save them to disks and only load the respective parts when needed.

Additional Information for Kleister-NDA It is worth recalling that the evaluation of it is tricky if using the provided official GEval evaluation script (Graliński et al., 2020)[13]. In detail, given the predited tokens, one has to retrieve the associated texts in a group. For example, the beginning of an entity group usually starts with a class beginning with "B-", followed by a series of "I-" tokens. However, there is no guarantee that the prediction will always return a group having this meaningful pattern, let alone many other complicated cases that can happen. Such complications make the post-processing of the prediction– before feeding to GEval–very difficult and importantly, not easily reproducible. In fact, amongst recent papers that report performance on this dataset (e.g. in Xu et al. (2020a); Appalaraju et al. (2021)), there is no published code provided.

[13] https://github.com/applicaai/kleister-nda

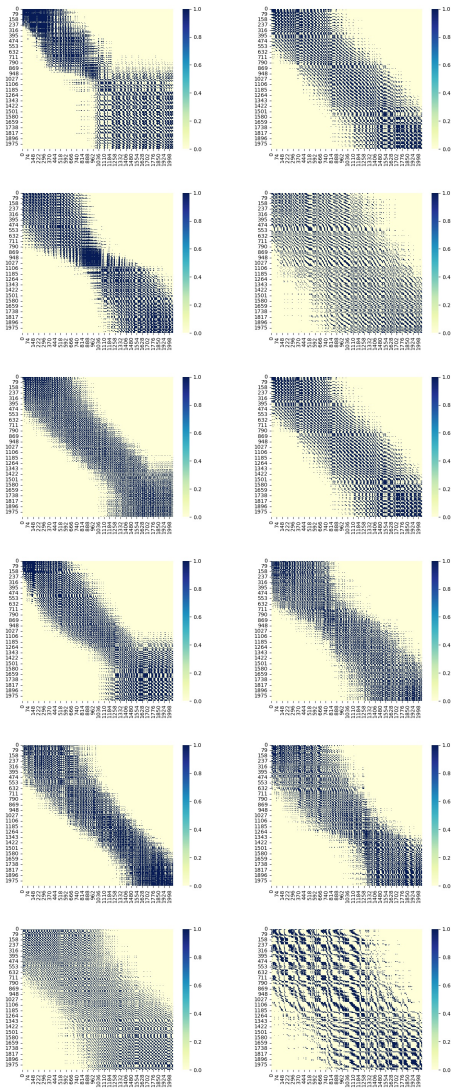


Figure 5: More distance masks from RVL-CDIP samples with the limit length of 2048 and 512 neighbors each.

Consequently, we treat this dataset the same as FunSD, given their similarity in annotations. In addition, because this dataset is larger and much more difficult (due to decoying texts) compared to FunSD, we analyze the train dataset and employ the weighted loss based on the distribution the given labels. As a result, our method is more transparent and reproducible.

C Additional Samples on Distance Masks

Complementary to Figures 2c and 2d, we present some more distance masks based on real samples taken from RVL-CDIP with the same setting in Figure 5.