LAMPReT: Layout-Aware Multimodal PreTraining for Document Understanding

Te-Lin Wu[*1], **Cheng Li**[2], **Mingyang Zhang**[2], **Tao Chen**[2],
Spurthi Amba Hombaiah[2], **Michael Bendersky**[2]
[1]University of California, Los Angeles, [2]Google Research
telinwu@cs.ucla.edu, {chgli,mingyang,taochen,spurthiah,bemike}@google.com

Abstract

Document layout comprises both structural and visual (*e.g.* font-sizes) information that are vital but often ignored by machine learning models. The few existing models which do use layout information only consider *textual* contents, and overlook the existence of contents in other modalities such as images. Additionally, spatial interactions of presented contents in a layout was never fully exploited.

To bridge this gap, we parse a document into content blocks (*e.g.* text, table, image) and propose a novel layout-aware multimodal hierarchical framework, LAMPReT, to model the blocks and the whole document. Our LAMPReT encodes each block with a multimodal transformer in the lower-level, and aggregates the block-level representations and connections utilizing a specifically designed transformer at the higher-level. We design hierarchical pretraining objectives where the lower-level model is trained with the standard masked language modeling (MLM) loss and the image-text matching loss, and the higher-level model is trained with three layout-aware objectives: (1) block-order predictions, (2) masked block predictions, and (3) image fitting predictions. We evaluate the proposed model on two layout-aware tasks – text block filling and image suggestion, and show the effectiveness of our proposed hierarchical architecture as well as pretraining techniques.

1 Introduction

Layout, the structural and visual presentation of the contents within a document, is a key aspect for writers to compose documents and for readers to understand documents. Specifically, the planning and the arrangements of how the contents are spatially structured, as well as the use of multiple modalities (*e.g.* texts, graphics, tables), is highly

influential in the choice of reading strategies from the readers. Hence, a well crafted layout can lead to better comprehension of the presented contents, and layout information is vital for document understanding (Wright, 1999; Hartley, 2013).

Learning a document-level representation with awareness of the layout has started to draw attention in the research community, especially in achieving better semantic document understanding (Katti et al., 2018; Denk and Reisswig, 2019; Xu et al., 2020). However, most prior works focus on rather surface forms of the layout, such as comprehending table hierarchy (Wang et al., 2020) and the claimed multimodality being referred to OCR detected features of the textual components (Hua et al., 2020; Zhang et al., 2020). Moreover, prior works mostly concern documents in the domain of scanned templated documents like receipts (Pramanik et al., 2020), not as *content-rich* and *layout-flexible* as articles such as Wikipedia pages.

In this paper, we propose **L**ayout-**A**ware **M**ultimodal **PreT**raining, dubbed LAMPReT, aiming for a more general-purposed pretraining methodology which exploits both the structure and the content of documents, and considers multimedia contents, such as images, to learn a comprehensive multimodal document representation. Specifically, we utilize an in-house document tokenizer to parse HTML formatted pages into several *content blocks*, where each *block* has the following features: (1) spatial position, (2) semantic types, *e.g.* headers and tables, and (3) attributes like font-sizes.

Inspired by the inherent hierarchy in the contents, our LAMPReT framework is hierarchical, consisting of two cascaded transformers (Vaswani et al., 2017). The lower-level transformer takes as inputs the parsed multimodal content blocks serialized by their sorted spatial positions, and the output *block-level* representations are consumed by the higher-level transformer. The lower-level model is trained with the Masked Language Modeling (MLM) ob-

---

jective (Devlin et al., 2019) and an image-to-text matching prediction for grounding different input modalities. For training the higher-level model, we propose three novel *block-level* pretraining objectives aiming to exploit the structure of a document: (1) **block-ordering prediction** requires the model to predict whether the input blocks are properly ordered, (2) **masked-block predictions** shares similar spirit with textual MLM but acts at the textual *block-level*, and (3) **image fitting predictions** requires the model to select the most suitable image for a missing image block.

We evaluate our proposed LAMPRET framework on two downstream document completion tasks: (1) **Text block filling** which aims to select the most appropriate textual block for a missing block to complete a document, and (2) **Image content suggestion** where the models are required to correctly retrieve the most suitable image at a layout position for a particular document, from a sizable set of candidates approximating realistic scenarios of composing documents. We show the effectiveness of our LAMPRET framework and the benefits of incorporating multimodality, as well as conduct extensive ablation studies on its components. Our main contributions are as follows:

- To our best knowledge, we are the first to consider layout multimodality in the context of the interactions between the actual image and the textual contents within a document.

- Propose a hierarchical framework with structure-exploiting pretraining objectives to learn layout-aware document representations.

- Design novel downstream tasks to evaluate the layout-awareness of the learned document representations, with the hope to spur relevant future research in multimodal document understanding.

2 Related Works

Document or Long-Text Learning. The recent advancements in NLP with the help of transformers (Vaswani et al., 2017), has encouraged research in transformer-based models that can go beyond the previous maximally allowed input length in models such as BERT (Devlin et al., 2019). Longformer (Beltagy et al., 2020), Reformer (Kitaev et al., 2020), and the recently proposed BigBird (Zaheer et al., 2020), are all capable of handling much longer input texts even to the whole document-level. The focus of this work, on one hand, is to design a framework that explicitly exploits the structure of the document contents, rather than modeling long document texts in the conventional way; on the other hand, we do not make any assumption of the base model used for LAMPRET and hence any of these recent models can replace our current base model, which is BERT.

Document Layouts. Obtaining document layouts can be done by utilizing a conventional and rather rule-based technique, such as VIPS (Cai et al., 2003), and recent deep learning approaches (Yang et al., 2017; Soto and Yoo, 2019; Ling and Chen, 2020), where the computer vision models (Ren et al., 2015) is adopted. CharGrid (Katti et al., 2018) and its extensions (Denk and Reisswig, 2019; Kerroumi et al., 2020) assume the layout contents are visually interpreted via computer vision techniques such as OCR, and propose learning frameworks to semantically understand the documents from a 2D aspect. Other prior works utilize document layouts as an effective component for information extractions (Hua et al., 2020; Gorai and Nene, 2020; Yu et al., 2020b), and provide a benchmark for surface semantic understanding of documents (Li et al., 2020a). Our work aims to go beyond surface understanding and exploit the layout explicitly in a modeling perspective in combined with the fine-grained language and vision models.

Multimodality. Multimodal grounding is an important paradigm for training visual-linguistics models. In this work, we adapt the basic model configuration from recently proposed BERT-based visual-linguistics models (Li et al., 2019; Su et al., 2020; Lu et al., 2019; Chen et al., 2020; Li et al., 2020b; Yu et al., 2020a) to fuse the textual and image modalities. Specifically, we also adapt the image-to-text matching training objective inspired by these models as one of the components in LAMPRET. Prior works concern multimodality in learning or extracting document layouts (Wang et al., 2020; Pramanik et al., 2020; Zhang et al., 2020; Xu et al., 2020) mostly by viewing a document as a structured imagery, while we model the actual multimodality in the contents such as how the texts should interact with the images within a document.

3 Preliminaries

Document Tokenizer: In this work, the layout is obtained by an in-house document parsing tool[1],

---

[1]Note that document parsing is not of our main focus, we assume our data is already parsed by the ready-to-use tool.

Figure 1: **An example page parsed by the document tokenizer.** Each red box indicates a content block. Blue colored coordinates are an exemplar block position tuple (origin of the coordinate system is at *top-left* corner of the entire document).

Expression	Descriptions
blk_i or blk_{ij}	The i-th content block in the serialized order, or the i and j-th in 2-dimension.
out_i	Final output representation of block i after the higher-level model.
blkh_i	Block-level representation of block i after the lower-level model, taken as input to the higher-level model.
CLS_i	CLS token for the i-th block, mainly for separating the input blocks and aggregating the i-th block-level representation.
global-CLS	The global CLS token which is prepended at the beginning of the inputs, where the representation obtained at this position is regarded as the overall representation of the whole document.
$\text{embd}_{feature}$	The embedding for a particular *feature*.

Table 1: **Terminologies** used throughout the paper.

which primarily handles HTML formatted webpage documents similarly to the aforementioned VIPS (Cai et al., 2003). Figure 1 illustrates how a document is *tokenized* (parsed) into several small *content **blocks***, each is a small proportion of the document which shows a clear spatial boundary to the others. Each block has the following features:

- **Block Position:** The 2D real valued position of the bounding box which encompasses the block, represented by XY coordinate tuples of (*top-left*, *bottom-right*) corners as illustrated in Figure 1. Each XY coordinate is normalized to $\in [0, 1]$.

- **Block Type:** The semantic type of the content presented in the block. There are in total 13 different types defined by our document tokenizer, with the most popular ones being header, paragraph, image, list (bullet-items), and table, etc.

- **Block Attributes:** The visual presentations of the **texts** featured in a block. Two types of attributes are generated by our tokenizer: (1) **scalar typed**, such as font-size, which is normalized to $\in [0, 1]$ with 1 indicating the largest possible font-size, and (2) **binary-typed**, such as indicating if the text is **bold**, *italic*, or underlined.

- **Multimedia:** There might be multimedia contents such as images, thumbnails, and videos in a block. In this work, we only consider images for our multimodal model. But our model can be easily extended to other multimedia contents. Note that multimedia content can be a block itself, as illustrated in Figure 1, the *Image* block is different from the *Image & Text* block.

Layout: We define layout as the structural presentation of the tokenized content blocks, *i.e.* their
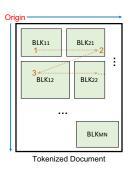
relative positions and orders, and the aforementioned attributional features of the textual contents within a block. In order to properly prepare the input representations for our models, we first **sort** the tokenized content blocks, with respect to the two dimensional coordinates of their **top-left** corners, as illustrated in Figure 2a. We sort the Y-axis first and then X-axis, with the intuition that for the documents this work focuses on, the vertical order is slightly more important than the horizontal. The sorted blocks are serialized in a **zigzag** fashion and then fed to the models. Note that our proposed way of sorting is not necessarily the optimal one, where our main focus here is to provide a meaningful ordering for the models to take inputs.
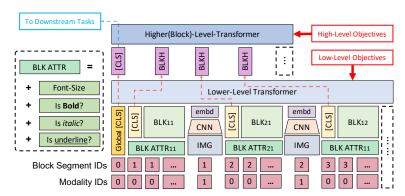
4 LAMPreT

Our LAMPRET framework aims to: (1) model the inherent hierarchical formulation of layout-structured documents, and (2) exploit the structure alongside the actual document contents to learn the representation. The frequently used terminologies throughout the paper are defined in Table 1.

4.1 Model Overview

Hierarchical Architecture: In order to better comprehend a document structured with a particular layout, we consider two level of layout hierarchical formulation. Specifically, the lower-level of the hierarchy refers to the contents of a block such as text and/or images, while the higher-level concerns how these blocks are spatially structured. We design a framework consisting of two cascaded transformers taking different levels of inputs of a given document, as illustrated in Figure 2b. The

(a) Layout Sorting.

(b) Hierarchical Architecture.

Figure 2: **LAMPRET Framework Overview: (a) Sorting and serializing blocks:** Setting the *origin* of the document at the top-left-most corner, we perform a 2D sorting on the content blocks anchoring around their *top-left* coordinate. We then serialize the sorted blocks in a *zigzag* fashion to obtain a reasonable ordering for the model inputs. **(b) Hierarchical formulation:** LAMPRET framework exploits the inherent hierarchical nature of document layouts. The input representation of each block blk$_i$ contains the embeddings of Wordpiece tokens, block-segment-ids, modalities, and attributional features. The output representations of the lower-level model at each CLS$_i$ position are fed to the higher-level model. Different levels of objectives are applied to the models in different hierarchy, and the representations at global-CLS are used in downstream tasks.

lower-level model takes as inputs the raw parsed contents, where each *content block* is placed at its **serialized sorted** position (represented by block-segment-id, recall Figure 2a). Each block contains the textual contents and potentially also a few images (can be more than one), making the lower-level model inherently multimodal. Each block blk$_i$ is prepended with a CLS$_i$[2] special token for indicating the boundary of block contents. We also prepend a global-CLS token at the beginning of the inputs for obtaining *document-level* representation. The higher-level model then takes as inputs the block-level representations blkh$_i$, *i.e.* the outputs of the lower-level model at each CLS$_i$ position.

Input Representations: The textual contents are tokenized by the WordPiece (Wu et al., 2016) tokenizer. Each block is attached with a **block-segment-id** *indexed by its serialized sorted position*, starting with 1 (0 is for the global-CLS position). We map and round each real-valued font-size to an integer $\in [0, 10]$. The **boldness**, underline, and *italic* are simply represented as binary values $\in \{0, 1\}$. We also supplement a binary embedding indicating the modality. The overall input representation for each token position is as follows:

$$\text{embd} = \text{embd}_{WordPiece} + \text{embd}_{block\_seg\_id}$$
$$+ \text{embd}_{type} + \text{embd}_{modality} + \text{embd}_{attr} \quad (1)$$

where the embd$_{attr}$ denotes the element-wise

summed embedding from all the textual attributes. For each block, we leave a design choice to truncate its contents with a maximally allowed token length, as well as a maximally allowed number of images. More details are in the appendix Section B.1.

Visual Embedding: The image contents are first fed to a convolutional neural network (CNN), followed by a transformation MLP (multi-layer perceptron) layer to align the resulting visual embedding embd$_{img}$ to the same size of the textual token embedding. For documents without any image contents, we simply pad the inputs with zero-image tensors, and the attention mask in the lower-level model is adjusted not to attend to those input positions. For those standalone image blocks, we attach them to the closest text paragraph block (determined by the block positions) for a more straightforward block-level representation aggregation.

4.2 Training Objectives

Figure 3 illustrates the training objectives on both the low- and the high-level of the LAMPRET framework. The lower-level training objectives aim to capture the finer-grained linguistics, visual information and the ability to handle multimodal inputs, while the higher-level training objectives aim to exploit the structural interactions among the contents at the *block-level*.

Low-level objectives for the lower-level model:

- **Masked Language Modeling (MLM):** Following BERT and its multimodal variants (Lu et al., 2019; Li et al., 2019), we apply the MLM objec-

---

[2]The CLS special token in Devlin et al. (2019) can be used for downstream tasks, we follow the similar formulation to prepend each block with its own CLS token so each of them can contribute to the training when required.
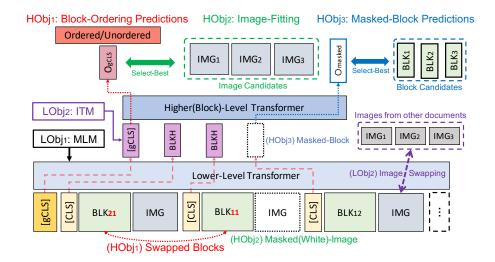
Figure 3: **LAMPreT Pretraining Objectives:** $HObj_i$ and $LObj_i$ denotes the i-th high- and low-level objective respectively. MLM and ITM stands for masked-language modeling and the image-text matching prediction for the low-level objectives. For each high-level objective, we illustrate: an exemplar block swapping for the block-ordering objective, an image masking for the image fitting objective, and a block masked at its *block-level representation* for the block-MLM objective, respectively.

tive on one hand to further finetune the linguistics ability on our dataset, on the other hand to fine-tune language modeling with image modality.

- **Image-Text Matching (ITM):** To further sharpen the model capability of handling multi-modal inputs, we adapt the image-text matching (ITM) prediction used in (Lu et al., 2019) to our setups. Specifically, for a given document d containing one or more images, we sample a few candidate images from other documents $\{d'\}$ within the same mini-batch during training, and swap them with some images in d with certain probabilities[3]. The model is then required to predict whether the textual contents match the resulting image sequences as binary classification.

High-level objectives for the higher-level model:

- **Block-Ordering Predictions (B-ORD):** Two input blocks are randomly selected and swapped[4] (with certain probability remained) in their serialized order when inputting to the the lower-level model. An MLP[5] which takes as input the output representation at the global CLS position, $out_{global-CLS}$, is trained to make the binary prediction on whether the input contents are following a proper order, *i.e.* whether the two selected

---

[3]With probability δ the images in d are swapped, and $1 - \delta$ the images in d are remained the same.

[4]We limit the random re-ordering of the blocks to 2 to leave majority of other blocks untouched for other training objectives, and it is also empirically proven sufficiently effective.

[5]We include the specifications of all the MLPs used in this work in the appendix section B.4.

blocks are swapped. The block-segment-ids for the two selected blocks are replaced by a padding value to prevent the leak of the original order.

- **Block-MLM (B-MLM):** One or more textual blocks are masked out at their *block-level* representations, $blkh_i$, by replacing them with zero-tensors. The objective requires the model to *select* the most suitable block for the masked position from a given set of candidate blocks, where the candidate set is constructed by collecting the blocks from all the documents (including self) within a mini-batch during training. An MLP layer then takes the concatenation of the output representations of the masked positions and the *block-level* representations of the candidate blocks, *i.e.* **concat**(out_{masked}, $blkh_i$, $blkh_j$, ...), and outputs the classification result of the index to the most suitable block. Since this objective is performed as a classification task, which requires a fixed number of candidates, in practice we truncate the candidates to a fixed number of blocks (with ground truth blocks deliberately included).

- **Image Fitting (IMG-FIT):** One or more images are masked out by replacing them with a *mask-image-token*, which is a white image in our implementation. This objective requires the model to select the most suitable images from a set of candidate images for the masked-out images. Similar to Block-MLM, the candidate set is constructed by collecting the images from all the documents within a mini-batch during

training, and a classification MLP layer is applied to predict the most suitable ones. The input to the MLP layer for the masked image in the i-th block is: **concat**(out$_{global-CLS}$, out$_{blk,i}$, embd$_{img,1}$, embd$_{img,2}$, ...), where embd$_{img,j}$ is the visual embedding of the j-th image candidate. We add the output representation at the global-CLS position to incorporate modeling the general trends of how the images are positioned within a document as it aggregates information from each block. In addition, a *batch-level* mask is applied to filter out the losses of those data entries (documents) without any image contents.

LAMPRET framework is jointly trained with a linear combination of the losses for the low- and high-level objectives:

$$L_{\text{LAMPRET}} = \lambda_1 L_{mlm} + \lambda_2 L_{itm} \\ + \lambda_3 L_{b-ord} + \lambda_4 L_{b-mlm} + \lambda_5 L_{img-fit} \quad (2)$$

where λ_is are tunable hyperparameters[6], and all the losses L are of classification cross-entropy loss.

5 Experiments

Our experiments aim to answer the following research questions: (1) Is the hierarchical formulation of LAMPRET effective? (2) Are the proposed layout-aware training objectives effective, and how are they complementing one another? (3) When and on what tasks does the multimodality help?

5.1 Evaluation Tasks

We design two *document completion* tasks for evaluating the models: (1) **text block filling** aims to evaluate the model capability of interpreting the structure of the textual contents, and (2) **image suggestion** concerns the layout multimodality in addition to the structural aspect on texts.

Text Block Filling: Suppose the 2D sorted and then serialized content blocks are $\{\text{blk}_1, \text{blk}_2, ..., \text{blk}_N\}$, we randomly select a block blk_i to mask, and provide the context $\text{blk}_{1:i-1}$ as inputs to the model, while leaving $\text{blk}_{j:j+K} \cup \text{blk}_i$ as candidates, where $j > i$, blk_j is spatially positioned after blk_i by a certain margin (four rows in a common web-page document). The closest K ($K = 5$) blocks abide by the criteria are selected to make this task challenging. The task is then to predict the correct block blk_i from the

---

[6]We set all λ_is to 1 in our actual implementations.

candidate blocks. Note that this masked block blk_i can be of any type of textual blocks, including header, an item from a list or table, etc. In this sense, the capability of prediction relies heavily on the understanding of the structure of the document.

Image Suggestion: The model takes as inputs all the content blocks of a document with an image masked-out (by replacing it with all-white-image), and is required to predict the correct image from a given set of candidate images (including the ground truth of the masked out one). We extract C ($C = 1000$ in this work) candidate images from documents unseen from the ones used during the pretraining for evaluating the models. Note that since this task aims to simulate suggesting the image contents when composing a novel document, for a more realistic setting, we **strip the textual blocks which encompass the direct captions to the images** in the dataset used for this task.

Finetuning on Downstream Tasks: To allow better fusion of low and high-level information, we have: $R_{doc} = \sigma(\alpha) \cdot \text{blkh}_{global-CLS} + (1 - \sigma(\alpha)) \cdot \text{out}_{global-CLS}$ to represent a document, where α is a task-dependent learnable scalar and σ is the Sigmoid function. We train an MLP to embed the document representation R_{doc} to retrieve from a set of candidate embeddings $\{R_{cand}\}$ with a contrastive loss (Hadsell et al., 2006), where $R_{cand,i} = \text{blkh}_i$ for text block filling and $R_{cand,i} = \text{embd}_{img,i}$ for image suggestion. Denote Y as equals to 1 if R_{doc} is paired with the ground truth R_{gt}, and 0 otherwise, and m a predefined margin, we have:

$$D_w(R_{doc}, R_{cand,i}) = \|\mathbf{MLP}(R_{doc}) - \mathbf{MLP}(R_{cand,i})\|_2$$
$$L_{\text{contrastive}} = (1 - Y)\frac{1}{2}(D_w)^2 + (Y)\frac{1}{2}\{max(0, m - D_w)\}^2$$
$$(3)$$

5.2 Baselines

We compare our LAMPRET framework to the following baselines, more details are in Section B.3:

Single-Level LayoutLM: The single-level, non-hierarchical variant of LAMPRET framework consists of only the lower-level model, which resembles the base model of the prior work LayoutLM (Xu et al., 2020). For training, both low-level objectives of LAMPRET, MLM and ITM, are utilized. We compare against this baseline for examining the effectiveness of the hierarchical formulation and the high-level objectives.

CNN-Grid: Inspired by prior work CharGrid and BERTGrid (Katti et al., 2018; Denk and Reisswig,

2019), we experiment replacing the transformer-based higher-level model with a CNN module (the original Char(BERT)Grid is applied on OCR detected results which does not fit our setup). Each *block-level* representation blkh_i is inserted to a position on a 2D map according to the sorted 2D coordinates of the block. This results in a 3D tensor, where the original 1D blkh_i representation becomes the channel dimension, and hence can be the input to a CNN module. While the output representation $\text{out}_{global-CLS}$ of LAMPRET acts as the overall representation of the entire document, we apply an average pooling at the output of the CNN module to obtain the document-level representation of the same size. Since the CNN is viewed as a substitute of the higher-level model, we train this baseline with the same set of objectives of LAMPRET, and hence the CNN-Grid baseline is the enhanced version of the prior works as we train it with additional layout-aware objectives.

Unimodality (Text-Only): We are also interested in examining how our framework performs particularly for the text block filling task, if **the textual contents are the only** model inputs. We experiment the text-only variant of models for both our LAMPRET and the single-level LayoutLM.

5.3 Implementation Details

We initialize all the lower-level models (note that LayoutLM baseline is single-level) with BERT-base-uncased pretrained weights released from the original authors. A pretrained CNN module is adopted for all the models and baselines to encode the images, and then transformed to the same embedding size of the Wordpiece token embedding, 768, with an MLP layer. For both block-MLM and image fitting pretraining objectives, we empirically select 20 for the size of the candidate set, and hence the final output projection layers for these two objectives are 20-way classification. For block-ordering, the output projection MLP is simply performing binary classification. More implementation details are in the appendix Section B.

5.4 Dataset

In this work, we scrape a collection of English Wikipedia pages (Wikipages) for training and evaluating the models. Our Wikipage dataset is uniformly sampled (scraped) from the entire Wikipedia, which makes it **diverse** and **rich** in the *genres* and the *contents*. Furthermore, the

Wikipage dataset naturally features **rich structures** as well as different **modalities** of contents. We preprocess the collected Wikipages as described in section 3 to obtain content *blocks*. Table 2 summarizes the essential dataset statistics.

Type	Counts	
Total Pretraining Wikipages	6.2M	
Total Wikipages for Text Block Filling	30K	
Total Wikipages for Image Suggestion	28K	
Downstream Train / Val / Test	70% / 10% / 20%	
Type	**Mean**	**Std**
# Blocks in a Wikipage	93.51	231.31
# Images in a Wikipage	0.46	1.33
Document Token Length	1083.02	1963.98

Table 2: **General statistics of the dataset.** Note that the Wikipages used in the downstream tasks are disjoint from the pretraining set. For the image suggestion task, we only retain pages with at least one image.

5.5 Evaluation Metrics

Since the downstream tasks are trained with contrastive objective and performed in a retrieval fashion, we adopt two common ranking-based metrics to quantify the model performances:

Mean Reciprocal Rank (MRR): We compute the reciprocal (*i.e.* the multiplicative inverse) ranks of the ground truth items in the given candidates list, and average them across the whole test set.

Recall @ K: We compute the recall in the top-K ranked items by counting the number of the ground truth items in such a top-K candidate list. Since we only have one ground truth item for each example, the recall is binary existence divided by K.

5.6 Experimental Results

Table 3 summarizes the model performances on the two proposed downstream tasks.

Text Block Filling. For the text block filling downstream task, our proposed LAMPRET model outperforms the baselines in both the precision and F-1 score metrics. It is worth noting that for both LAMPRET and the single-level LayoutLM, the unimodal text-only version performs slightly better than the multimodal version. We hypothesize that such results can be attributed to the suboptimal multimodal representation fusing, that it can be potentially alleviated with more sophisticated and finer-grained multimodal grounding paradigms. Among the models, CNN-Grid baselines performs the worst, of which the attention mechanism in transformers is hypothesized to capture the block-level interactions better.

Method	Modality	Text Block Filling			Image Suggestion (C=1000)	
		F1-Score	Precision (%)	Recall (%)	Recall @ 5 (%)	MRR (%)
CNN-Grid	Multimodal	39.92	40.45	39.40	67.33	62.60
Single-Level LayoutLM	Text-Only	51.49	39.93	**72.45**	—	—
	Multimodal	51.30	41.40	67.43	75.99	76.54
LAMPRET	Text-Only	**52.36***	**42.37***	68.50	—	—
	Multimodal	52.09*	41.85*	68.98	**99.98***	**98.55***

Table 3: **Model Performances:** Best performances for each metric is boldfaced. Our LAMPRET framework outperforms the carefully crafted baseline models in both tasks. *** indicates that our LAMPRET framework is statistically significant compared with the best-performing baselines, according to the size of our test-set at level of 0.01.**

Ablated Components	Text Block Filling			Image Suggestion	
	F1-Score	Precision (%)	Recall (%)	Recall @ 5 (%)	MRR (%)
w/o Image Fitting Objective	50.72	38.82	39.40	90.63	89.68
w/o Block-MLM	51.18	41.46	66.86	86.11	85.26
w/o block-ordering	50.19	37.78	74.76	69.57	69.51
w/o Layout-Attributes	50.98	40.39	69.07	96.42	95.27
Multimodal LAMPRET	52.09	41.85	68.98	99.98	98.55

Table 4: **Model Ablation Studies:** We examine the contribution of the high-level pretraining objectives on the multimodal version of LAMPRET. Each row denotes the pretraining is conducted with the indicated objective excluded. We also include an ablation on pretraining without the attributional features (last row), which is shown more effective on the text-based task.

Image Suggestion. We only conduct the image suggestion downstream task for multimodal models, as unimodal ones do not have access to image based features. Our LAMPRET achieves almost perfect performance for both metrics (99%), while all the baseline models suffer significant performance degradation. The hierarchical formulation and the accompanying high-level objectives are proven effective in LAMPRET as compared to single-level LayoutLM. The CNN-Grid baseline again generally performs the worst.

Model Ablation Studies. We are interested in analyzing the contributions of the high-level pretraining objectives for different downstream tasks. Table 4 shows an ablation analysis on the multimodal version of our LAMPRET framework. At each row, we deduct (1) one of the high-level pretraining objectives, or (2) the layout attributional features. In general, the block-ordering objective is empirically proven quite effective for both downstream tasks, judged by the performance degradation when it is excluded during the pretraining. It is worth noting as well that the exclusion of the layout attributes do not cause that much deterioration compared to other pretraining objectives, which hypothetically implies that the exploitation of the structural information of layout designed in the proposed LAMPRET framework is relatively more effective.

Here we want to point out that we train for much more iterations during downstream finetuning for the image suggestion task (until performance convergence) when the model is not pretrained with the image fitting objective, otherwise with the same settings this downstream task does not work.

6 Conclusions and Future Works

We propose a multimodal layout-aware document representation learning framework, LAMPRET. Once a document is parsed into several spatially structured *content blocks*, we sort and serialize them in a 2D formulation. LAMPRET aims to model the inherent hierarchical formulation of a document layout by two cascaded transformers. The lower-level model is trained with MLM and ITM objectives, while the higher-level model is trained with three specifically designed layout-exploiting objectives. We evaluate LAMPRET on two downstream tasks: (1) text block filling, and (2) image suggestion task.

For future works, we envision that (1) a more delicately designed training paradigm such as curriculum training for our hierarchical models can be beneficial, and (2) a more sophisticated multimodal grounding to the lower-level model can potentially alleviate the slightly worse performance in the text-based downstream tasks. Furthermore, we hope to expand the target domains from Wikipages to other *content-rich* documents such as news articles.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283.

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

Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. 2003. Vips: a vision-based page segmentation algorithm.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Uniter: Learning universal image-text representations. In *European Conference on Computer Vision (ECCV)*.

Timo I Denk and Christian Reisswig. 2019. Bertgrid: Contextualized embedding for 2d document representation and understanding. In *Workshop on Document Intelligence at NeurIPS 2019*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 4171–4186.

Manoj Gorai and Manisha J Nene. 2020. Layout and text extraction from document images using neural networks. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 1107–1112. IEEE.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

James Hartley. 2013. *Designing instructional text*. Routledge.

Yuan Hua, Zheng Huang, Jie Guo, and Weidong Qiu. 2020. Attention-based graph neural network with global context awareness for document understanding. In *Proceedings of the 19th Chinese National Conference on Computational Linguistics*, pages 853–862.

Anoop R Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. 2018. Chargrid: Towards understanding 2d documents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4459–4469.

Mohamed Kerroumi, Othmane Sayem, and Aymen Shabou. 2020. Visualwordgrid: Information extraction from scanned documents using a multimodal approach. *arXiv preprint arXiv:2010.02358*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations (ICLR)*.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*.

Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. 2020a. Docbank: A benchmark dataset for document layout analysis. In *International Conference on Computational Linguistics (COLING)*.

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

Meng Ling and Jian Chen. 2020. Deeppapercomposer: A simple solution for training data preparation for parsing research papers. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 91–96.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 13–23.

Subhojeet Pramanik, Shashank Mujumdar, and Hima Patel. 2020. Towards a multi-modal, multi-task learning based pre-training framework for document representation learning. *arXiv preprint arXiv:2009.14457*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Carlos Soto and Shinjae Yoo. 2019. Visual detection with context for document layout analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3455–3461.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. Vl-bert: Pre-training of generic visual-linguistic representations.

In *International Conference on Learning Representations (ICLR)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Zilong Wang, Mingjie Zhan, Xuebo Liu, and Ding Liang. 2020. Docstruct: A multimodal method to extract hierarchy structure in document for general form understanding. In *Findings of Empirical Methods in Natural Language Processing (EMNLP)*.

Patricia Wright. 1999. The psychology of layout: Consequences of the visual structure of documents. *American Association for Artificial Intelligence Technical Report FS-99-04*, pages 1–9.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pretraining 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.

Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C Lee Giles. 2017. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5315–5324.

Fei Yu, Jiji Tang, Weichong Yin, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020a. Ernievil: Knowledge enhanced vision-language representations through scene graph. *arXiv preprint arXiv:2006.16934*.

Wenwen Yu, Ning Lu, Xianbiao Qi, Ping Gong, and Rong Xiao. 2020b. Pick: Processing key information extraction from documents using improved graph learning-convolutional networks. *arXiv preprint arXiv:2004.07464*.

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. In *Advances in Neural Information Processing Systems*, volume 33.

Peng Zhang, Yunlu Xu, Zhanzhan Cheng, Shiliang Pu, Jing Lu, Liang Qiao, Yi Niu, and Fei Wu. 2020. Trie: End-to-end text reading and information extraction for document understanding. In *ACM International Conference on Multimedia (ACMMM)*.

A More Details of The Layout

We hereby include more details that could help understanding the concept of layout defined in this work and how to process it, as well as its essential building component, the *content blocks*.

A.1 Sorting The Blocks

In our dataset, the block positions are structured in the form of a standard bounding box coordinate tuple: $(X_{\text{left}}, Y_{\text{top}}, X_{\text{right}}, Y_{\text{bottom}})$, where the pair $(X_{\text{left}}, Y_{\text{top}})$ represents the coordinates of the *top-left* corner, while the pair $(X_{\text{right}}, Y_{\text{bottom}})$ represents the *bottom-right* corner of the bounding box of a particular content block. In this formulation, even non-rectangular bounding regions for contents can be properly represented and bounded, and our sorting paradigm does not overlook those blocks.

We sort the blocks anchoring around the *top-left* corner of the box bounding the content block, in this way it is ensured their starting position is of the main consideration. Since even a semantically latter block can have earlier ending position if viewed from the document origin as illustrated in Figure 2a, we refrain from using the *bottom-right* corner for such a reason. We sort the two coordinates following the order: sort the Y_{top} first and then X_{left}, with a standard sequence sorting algorithm. We then perform a *zigzag* traversal to serialize the content blocks for the inputs to the models. Note that if the blocks are properly sorted (such as the aforementioned sorting), the serialization would reasonably preserve the relative orders in their original 2D formulations, and hence is a proper input sequence to a transformer-based model. The block-segment-ids defined after the serialization then acts similarly to the positional encoding of a transformer model, only that it is at the *block-level*.

A.2 Block Features

Block Attributes: For the real-value-typed textual attributional features, *i.e.* font-size, our original data is prepared with a normalized value with the maximum indicating the H1 header size in a standard HTML web-page. We map this to a range of $\in [0, 10]$, and round the results to obtain integer values, so that we can regard these integer-scalars as font-size ids and use the standard embedding technique, *i.e.* a linear layer (matrix) which retrieves the 1D embedding with the input ids.

Block Types: Each of the 13 block types is attached with an integer id which can then be used

Feature	Ranges or Bounds
Block Type	[0, 13]
Font-Size	$\{0, 1\}$, Binary
Is Bold	$\{0, 1\}$, Binary
Is Italic	$\{0, 1\}$, Binary
Is Underline	$\{0, 1\}$, Binary
Modality	$\{0, 1\}$, Binary
Block-Segment-ID	[0, max. # of blocks]
WordPiece Vocabulary	[0, 30521]

Table 5: **Ranges of discrete features:** For all the discrete and/or discretized features (*e.g.* font-size), we include their ranges or bounds in this table. For the maximum number of blocks used in this work, please refer to Table 7.

for standard embedding technique, starting from 0. We use block type id = 13, which is the 14-th id for indicating padding block, which is used in the block-ordering prediction objective for preventing leaking order information, as mentioned in section 4.2. We include Table 5 to summarize the ranges described above for all the discrete and/or discretized features of the inputs to our models.

B More Implementations Details

We implement all of the models (including all the baselines) in TensorFlow 1 (Abadi et al., 2016). The BERT-base models which all the models based on, is adapted from the codes released in the original author code repository. We also use the provided pretrained weights to initialize all of the models, specifically the lower-level models for those hierarchically formulated models. The vocabulary size of the BERT-base model is 30522 according to the original configurations, where the input are tokenized by the WordPiece tokenizer.

B.1 Input Representations

In section 4.2 in the main paper, we omit one of the obvious components used in the original BERT model, that is the token-level positional encoding, *i.e.* embd$_{positional}$. Particularly for this positional embedding, we simply use a consecutive positional id scheme for all the inputs at their *token-level*, starting from the first content block to the last and viewing the entire document contents as a single piece sentence segment. The image contents are also attached with a positional id without breaking the consecutive nature, we do not use different positional encoding schemes for different modalities and view them as the same piece of input for this encoding. As a result, the overall input representa-

tion is as follows:

$$\text{embd} = \text{embd}_{WordPiece} + \text{embd}_{block\_seg\_id}$$
$$+ \text{embd}_{type} + \text{embd}_{modality} + \text{embd}_{attr} \quad (4)$$
$$+ \text{embd}_{positional}$$

B.2 Higher-Level Model

For our LAMPRET framework, the architecture of our higher-level model is a N-layered transformer encoder, where the configuration of each layer of the encoder shares the same configurations used in BERT (where in BERT-base model, there are 12 this kind of layers cascaded sequentially). In our actual implementations, we use $N = 3$ for our higher-level model. All the other hyperparameters are remained the same as those in 1-layer BERT encoder (identical architecture for all 12 layers in the original BERT), where the hidden size is 768 and the number of attention heads is 12.

B.3 Baseline Models

B.3.1 CNN-Grid

Figure 4 illustrates the CNN-Grid baseline described in section 5.2 in the main paper. As can be seen, the lower-level model is identical to our proposed main framework LAMPRET, while we use a convolutional architecture for the higher-level model instead of the (empirically proven better) transformer. For each *block-level* representation blkh$_i$, we *insert* it to a 2D map according to its sorted 2D XY-position, with the dimension of blkh$_i$ being the third dimension of the 2D map, which eventually results in a 3D tensor. As a result, the constructed tensor should have the size of: (Max. number of block, Max. number of block, Dimension of blkh$_i$), *e.g.* $(50, 50, 768)$ in this work. We then apply a L-layered CNN module to encode this 3D tensor, with kernel size = 3 and $L = 3$ to match the $N = 3$ of the higher-level model of LAMPRET. We adjust the kernel strides and paddings in the CNN module to ensure that the height and width of the encoded tensor remains same as the input 3D tensor, in this way we can serialize back the encoded 3D tensor following the same $zigzag$ fashion to perform the high-level pretraining objectives in LAMPRET. However, since the CNN module is fundamentally different from the transformer model, we do not have a global-CLS position, instead, we apply an average pooling to the outputs of the CNN module to reduce the height and width of the final encoded tensor to 1×1, a 1D representation as the

Description	Output Dimension
Transformation layer for visual embedding	768
MLP for ITM prediction	2, binary
MLP for Block-ordering	2, binary
MLP for Block-MLM	20, classification
MLP for Image-Fitting	20, classification
MLP for downstream tasks	64

Table 6: **Configurations of all the MLPs in this work:** The description indicates where the MLP layers are being used. We implement all of our MLP layers as one layered MLP and include the output dimension in this table.

higher-level document-level representation similar to out$_{global-CLS}$. In combined with the aforementioned serialized back output representations, we can apply all the high-level objectives similar to our LAMPRET framework.

B.3.2 Single-Level LayoutLM

Although we call this baseline as *LayoutLM*, it is still substantially different from the proposed model in (Xu et al., 2020). The original LayoutLM utilizes an OCR-based vision parser on scanned documents, and hence the attributional visual presentations of the textual contents, such as the font-sizes and boldness, are implicitly embedded in the OCR-parsed contents. On the contrary, in our version of Single-Level LayoutLM, the block information and attributes are directly extracted from HTML elements, where the visual presentations of those attributes are explicitly encoded as several discrete variables. Another noticeable difference is that the original LayoutLM adopts 2D scalar-valued positional encoding from the four coordinates of the bounding boxes, while we first properly define a 2D sorting scheme and then serialize the content blocks in our layout so we only need to encode single-scalar block-segment-ids as the block-level positional encoding, with the fine-grained token-level positional encoding kept. Differing from the original LayoutLM, the MLM applied to our version of Single-Level LayoutLM is the standard linguistics-based masked learning, and we additionally incorporate the image-to-text matching prediction for aligning multimodal inputs, which is not concerned by the original LayoutLM work.

B.4 MLP Layers

Table 6 summarizes the configurations of all the MLP layers in this work. We list each MLP module mentioned in the main paper for better references on the implementation details.
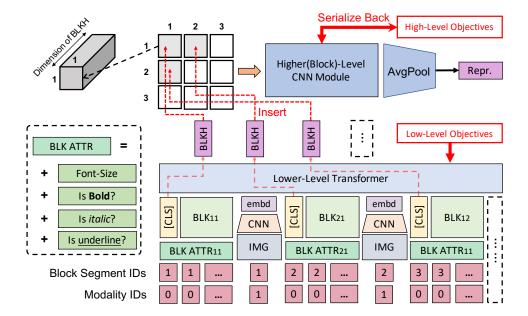
Figure 4: **Illustration of CNN-Grid baseline:** The block-level representations $blkh_i$ are inserted to the corresponding 2D positions of a 2D map, *e.g.* $blkh_{21}$ of blk_{21} is inserted to the position $(2, 1)$ (X-axis is vertical and Y-axis is horizontal). This will result in a 3D tensor since as the dimension of $blkh_i$ will be the *channel dimension* of the inserted map. This 3D tensor is then fed to the convolutional modules, where the CNN module (represented as the rectangular shape) is designed to output the same shape of the input 3D tensor. Notice that the global-CLS is not added since we do not need it for CNN higher-level model, instead, we perform an average pooling for reducing the height and width of the output tensor to 1×1 for obtaining a 1D *document-level* representation. The high-level objectives can be applied to the combination of the *document-level* representation and each of the out_i after serializing the output tensor of the first stage CNN module back.

C   More Training Details

C.1   Hyperparameters

All the essential hyperparameters used throughout this work can be referred to in Table 7. We also include the search bounds as well as the number of trials in searching for our manually-tuned hyperparameter search procedures in Table 8.

C.2   Hardware and Run-time

We train all of our models (including all the baselines) on a set of TPU computing hardwares, similarly to the configuration for the original BERT model, *i.e.* 4 Cloud TPUs in Pod configuration (16 TPU chips in total). The run-time for the pretraining phase on average takes 2-3 days, and the run-time for the two downstream tasks take roughly 4 hours each.

Method	Phase	Batch-Size	Initial LR	# Training Iterations	Max. Token Length	Max. Per-Block Token Length	Max. # blocks	# Params
CNN-Grid	Pretraining	128	2×10^{-5}	600K	512	50	50	120M
	Text Block Filling	128	1×10^{-6}	30K	512	50	50	
	Image Suggestion	128	1×10^{-6}	10K	512	50	50	
Single-Level LayoutLM	Pretraining	128	2×10^{-5}	600K	512	50	50	110M
	Text Block Filling	128	1×10^{-6}	30K	512	50	50	
	Image Suggestion	128	1×10^{-6}	10K	512	50	50	
LAMPReT	Pretraining	128	2×10^{-5}	600K	512	50	50	140M
	Text Block Filling	128	1×10^{-6}	30K	512	50	50	
	Image Suggestion	128	1×10^{-6}	10K	512	50	50	

Table 7: **Hyperparameters used for each model during different phases of training:** *Initial LR* denotes initial learning rate. All the models are trained with Adam optimizers (Kingma and Ba, 2015). We include number of parameters of each model in the last column, denoted as *# params*. We use the same set of hyperparameters for the models involving different versions when using different modality of inputs. Particularly for the image suggestion downstream tasks, the models without the corresponding image-fitting pretraining objectives in the ablation studies in Table 4, we train for 100K iterations, 10X the iterations used for those with the image-fitting pretraining objective, *i.e.* 10K iterations as shown above. For the learning rate decay, we follow the procedure provided by the original BERT implementation.

Type	Batch Size	Initial LR	# Training Iterations	Max. # blocks
Bound (lower–upper)	32–256	5×10^{-5}–1×10^{-6}	100K—600K	20—50
Number of Trials	6–10	2–3	2–4	8—10

Table 8: **Search bounds for hyperparameters:** for the hyperparameters of all the models.