XDoc: Unified Pre-training for Cross-Format Document Understanding

Jingye Chen[*], **Tengchao Lv, Lei Cui, Cha Zhang, Furu Wei**
Microsoft Corporation
{v-jingyechen,tengchaolv,lecu,chazhang,fuwei}@microsoft.com

Abstract

The surge of pre-training has witnessed the rapid development of document understanding recently. Pre-training and fine-tuning framework has been effectively used to tackle texts in various formats, including plain texts, document texts, and web texts. Despite achieving promising performance, existing pre-trained models usually target one specific document format at one time, making it difficult to combine knowledge from multiple document formats. To address this, we propose XDoc, a unified pre-trained model which deals with different document formats in a single model. For parameter efficiency, we share backbone parameters for different formats such as the word embedding layer and the Transformer layers. Meanwhile, we introduce adaptive layers with lightweight parameters to enhance the distinction across different formats. Experimental results have demonstrated that with only 36.7% parameters, XDoc achieves comparable or even better performance on a variety of downstream tasks compared with the individual pre-trained models, which is cost effective for real-world deployment. The code and pre-trained models will be publicly available at https://aka.ms/xdoc.

1 Introduction

Document understanding has undoubtedly been an important research topic as documents play an essential role in message delivery in our daily lives (Cui et al., 2021). During the past several years, the flourishing blossom of deep learning has witnessed the rapid development of document understanding in various formats, ranging from plain texts (Devlin et al., 2018; Liu et al., 2019; Dong et al., 2019), document texts (Xu et al., 2020, 2021a; Huang et al., 2022), and web texts (Chen et al., 2021; Li et al., 2022a; Wang et al., 2022b). Recently, pre-training techniques have been the de facto standard for document understanding, where the model is first pre-trained in a self-supervised manner (e.g. using masked language modeling as the pretext task (Devlin et al., 2018)) with large-scale corpus, then fine-tuned on a series of downstream tasks like question-answering (Rajpurkar et al., 2016; Mathew et al., 2021), key information extraction (Jaume et al., 2019; Xu et al., 2022) and many others. Albeit achieving impressive performance on specific tasks, existing pre-trained models are far from flexible as they can only tackle texts in a single format (e.g. LayoutLM (Xu et al., 2020) is designed for tackling document texts and is not suitable for web texts). This makes it difficult to combine knowledge from multiple document formats. Meanwhile, the category of pre-trained models will keep increasing if more formats (e.g. Word and PowerPoint) are further studied in academia.

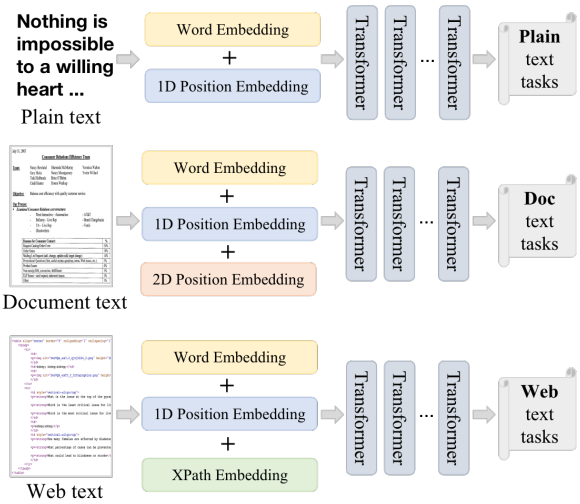
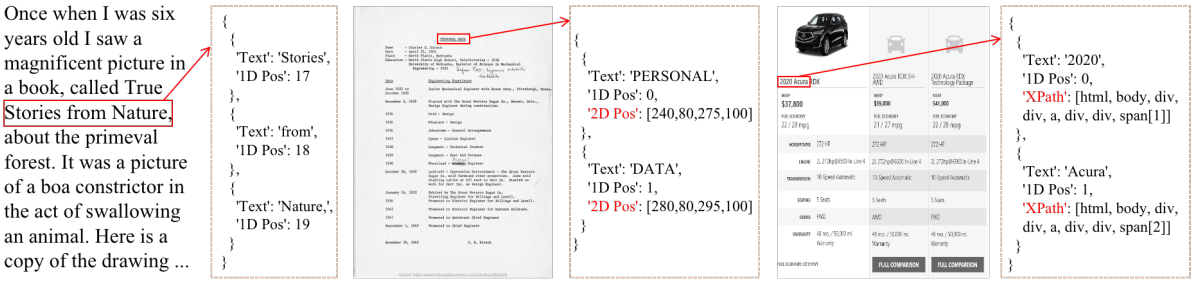


Figure 1: Pre-trained models for different document formats. Most of the structures are similar (word embedding, 1D position embedding, and Transformer layers) while only a small proportion of the structures (2D position and XPaths embedding) are different.

Among different pre-trained models for document understanding, it is observed that many pre-

[*] Work done during internship at Microsoft Research Asia.



(a) An illustration of plain text (b) An illustration of document text (c) An illustration of web text

Figure 2: Illustrations of three document formats. For each format, the corresponding meta-information is shown in the dash boxes. Please note that the text content and 1D position are common attributes across three formats while 2D position and XPath strings (marked as red) are specific for document and web texts respectively.

trained models share a similar architecture, such as a word embedding layer, a 1D position embedding layer, and Transformer layers (see Figure 1). In contrast, there are also different parts serving as prior knowledge for a specific format (e.g. two-dimensional coordinates for document texts and XPaths for web texts). Intuitively, we find that the parameters of different parts are far less than the parameters of the shared backbones. For instance, $\text{LayoutLM}_{\text{BASE}}$ (Xu et al., 2020) based on RoBERTa (Liu et al., 2019) consists of 131M parameters while the 2D position embedding layer only contains 3M parameters (2.3%). Similarly, $\text{MarkupLM}_{\text{BASE}}$ (Li et al., 2022a) based on RoBERTa has 138M parameters while the XPath embedding layer only contains 11M parameters (8.0%). Therefore, it is indispensable to design a unified pre-trained model for various text formats while sharing backbone parameters to make models more compact.

To this end, we propose XDoc, a unified architecture with multiple input heads designed for various categories of documents. For the sake of parameter efficiency, we share the backbone network architecture across different formats, including the word embedding layer, the 1D position embedding layer, and dense Transformer layers. Considering that the different parts only take up a small proportion in XDoc, we introduce adaptive layers to make the representation learning for different formats more robust. We collect the large-scale training samples for different document formats, and leverage masked language modeling to pre-train XDoc. Specifically, we use three widely-used document formats for experiments, including plain, document, and web texts (see Figure 2 for more details). To verify the model accuracy, we select the GLUE benchmark (Wang et al., 2019) and SQuAD (Rajpurkar et al., 2016, 2018) to evaluate plain text understanding, FUNSD (Jaume et al., 2019) and DocVQA (Mathew et al., 2021) to evaluate document understanding, and WebSRC (Chen et al., 2021) for web text understanding. Experimental results have demonstrated that XDoc achieves comparable or even better performance on these tasks while maintaining the parameter efficacy.

The contributions of this paper are summarized as follows:

- We propose XDoc, a unified pre-trained model that tackles texts in various formats in pursuit of parameter efficiency.
- Pre-trained with only masked language modeling task, XDoc achieves comparable or even better accuracy on various downstream tasks.
- The code and pre-trained models will be publicly available at `https://aka.ms/xdoc`.

2 XDoc

In this section, we first introduce the architecture of XDoc and details of the embedding used for each document format, then introduce the objectives for pre-training the XDoc model.

2.1 Model Architecture

As is demonstrated in Figure 3, XDoc is capable of tackling texts in various formats (plain, document, and web texts) in one model. For any input sequences, XDoc learns to embed them using a shared backbone and additional embedding layers when other prior knowledge is available. In detail, for any input text T, XDoc first tokenizes it

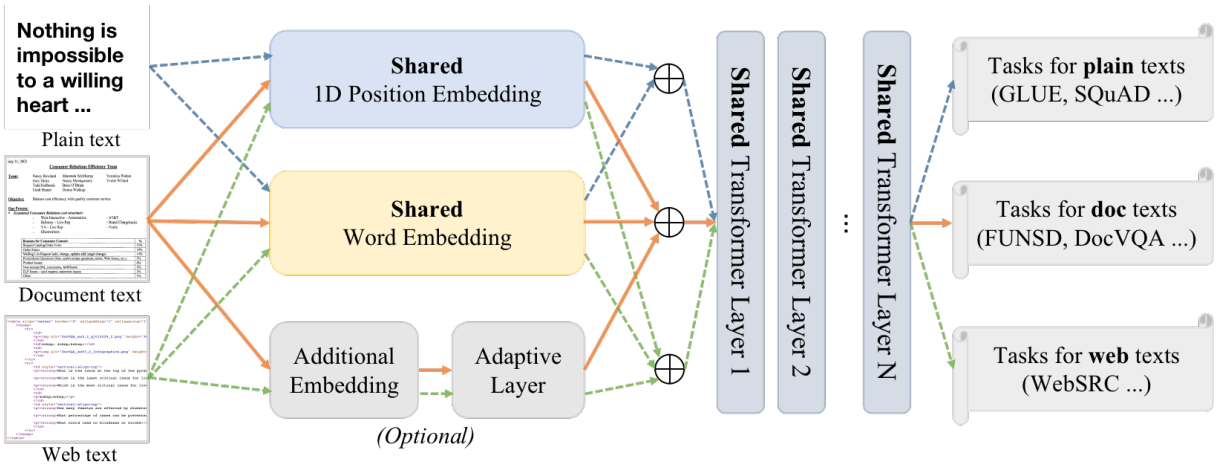


Figure 3: XDoc tackles multiple formats in one model while sharing most parameters including 1D position embedding, word embedding, and dense Transformer layers. An *optional* embedding layer and adaptive layer are utilized for specific prior knowledge such as 2D position for document texts and XPaths for web texts (no additional prior for plain texts). We demonstrate the dataflow for document texts and use *dash* lines for other formats.

into subwords $\mathbf{s} = s_{1:L}$ using WordPiece, where L denotes the maximum length. Subsequently, for each subword s_i with index i, it is first fed to a word embedding layer and we denote the output as $\mathrm{WordEmb}(s_i)$. Then it is added with a learnable 1D position embedding $\mathrm{1DEmb}(i)$. Since the word embedding and 1D position embedding layers are indispensable for Transformer-based models, we attempt to **share** the parameters across different formats. Based on this, we will detail the overall embedding for each document format in the next.

Overall embedding for plain texts As there is no additional prior knowledge for plain texts, we simply add up the word embedding and 1D position embedding to construct the input for Transformer layers following (Devlin et al., 2018; Liu et al., 2019). For each word s_i^P, where i is the index and "P" denotes "Plain", the overall embedding $\mathrm{Emb}(s_i^P)$ can be calculated as follows:

$$\mathrm{Emb}(s_i^P) = \mathrm{WordEmb}(s_i^P) + \mathrm{1DEmb}(i) \quad (1)$$

Overall embedding for document texts Different from the plain texts, the visually rich document texts are usually organized with 2-D layouts, where the coordinates of each text box play crucial roles in understanding. Hence, the 2D position should be necessarily taken into account during pre-training. Concretely, for a given subword s_i^D ("D" is the abbreviation of "Document"), we denote the 2D position as $box_i^D = (l_i, r_i, t_i, b_i, w_i, h_i)$, where l, r, t, b, w, h denote left, right, top, and bottom coordinates, width and height of the text box, respectively. For example, as illustrated in Figure 2(b), l, r, t, b, w, h of the text "PERSONAL" is set to 240, 275, 80, 100, 35, and 20, respectively. Considering that most parameters are shared across different formats, we introduce an adaptive layer to enhance the distinction of specific prior information. The adaptive layer is simply implemented with a lightweight *Linear-ReLU-Linear* sequence and we will discuss the effectiveness in Section 3.4. Following (Xu et al., 2020, 2021a), we add up all the embedding to construct the overall embedding $\mathrm{Emb}(s_i^D)$ as follows:

$$\begin{aligned}\mathrm{Emb}(s_i^D) = \mathrm{WordEmb}(s_i^D) + \mathrm{1DEmb}(i) \\ + \mathrm{DocAdaptive}[\mathrm{2DEmb}(box_i^D)]\end{aligned} \quad (2)$$

$$\begin{aligned}\mathrm{2DEmb}(box_i^D) = \mathrm{LeftEmb}(l_i) + \mathrm{RightEmb}(r_i) \\ + \mathrm{TopEmb}(t_i) + \mathrm{BottomEmb}(b_i) \\ + \mathrm{WidthEmb}(w_i) + \mathrm{HeightEmb}(h_i)\end{aligned} \quad (3)$$

where "LeftEmb" denotes the embedding layer of the left coordinates (other embedding layers follow the same naming conventions). Please note that the adaptive layer is not shared across different formats and "DocAdaptive" is specifically used for document texts.

Overall embedding for web texts Since the 2-D layout of each website is not fixed and it highly

depends on the resolution of rendering devices, we only employ XPath as the prior knowledge following (Li et al., 2022a). Concretely, for each subword s_i^W ("W" is the abbreviation of "Web"), its XPath $xpath_i^W$ can be represented with a tag sequence and a subscript sequence. Taking the text "Acura" in Figure 2(c) as an instance, its original XPath expression is /html/body/div/a/div/div/span[2]. Following MarkupLM (Li et al., 2022a), we construct the tag sequence as [html, body, div, a, div, div, span], representing the tag order from the root to the current node. In addition, the subscript sequence is set to [0, 0, 0, 0, 0, 0, 2], where each subscript denotes the index of a node when multiple nodes have the same tag name under a parent node (More explanations are shown in Appendix A). We add the tag embedding and subscript embedding to get the XPath embedding $\mathrm{XPathEmb}(xpath_i^W)$. The overall embedding can be calculated as:

$$\begin{aligned}\mathrm{Emb}(s_i^W) &= \mathrm{WordEmb}(s_i^W) + \mathrm{1DEmb}(i) \\ &+ \mathrm{WebAdaptive}[\mathrm{XPathEmb}(xpath_i^W)]\end{aligned} \tag{4}$$

Similarly, we leverage an adaptive layer "WebAdaptive" for better pre-training. Further, the overall embedding is fed to **shared** Transformer layers to obtain the contextual representations.

2.2 Pre-training Objectives

We employ masked language modeling (MLM) as the pre-training task following (Devlin et al., 2018; Liu et al., 2019; Xu et al., 2020). More specifically, we randomly mask 15% of the input tokens, where 80% tokens are converted to a special [MASK] token, 10% tokens are randomly replaced with other tokens, and 10% tokens remain unchanged. Through pre-training, the model needs to maximize the probability of the masked tokens according to the contextual representations.

3 Experiments

In this section, we first introduce the model configuration and detail the hyperparameters in XDoc, then introduce the pre-training strategies of XDoc. Next, we demonstrate the experimental results on a wide range of downstream tasks. At last, we verify the effectiveness of some designs in XDoc and have a discussion.

3.1 Model Configurations

The proposed XDoc is initialized with $\mathrm{RoBERTa}_{\mathrm{BASE}}$, containing 12 Transformer layers, 768 hidden units, and 12 attention heads. The maximum length of each input sequence is set to 512 with a [CLS] token and a [SEP] token padding at the beginning and the end, respectively. The input sequence whose length exceeds 512 will be truncated, while the sequence shorter than 512 will be padded with [PAD] tokens.

3.2 Pre-training XDoc

Large quantities of corpus play an essential role in learning robust representations during pre-training (Liu et al., 2019). Specifically, we utilize three categories of datasets for pre-training, which are detailed as follows.

Pre-training data for plain texts. We follow (Liu et al., 2019) to leverage five English-language corpora for pre-training, including BOOKCORPUS (Zhu et al., 2015), English WIKIPEDIA[1], CC-NEWS (Nagel, 2016), OPENWEBTEXT (Aaron Gokaslan, 2019), and STORIES (Trinh and Le, 2018), totaling 213,713 files for pre-training.

Pre-training data for document texts. We leverage the large-scale scanned document image data IIT-CDIP Test Collection 1.0 (Lewis et al., 2006) following (Xu et al., 2020, 2021a; Huang et al., 2022). This dataset contains 42 million document pages, each of which is processed by OCR tools Tesseract[2] to yield the text contents and locations. For a fair comparison with previous works, we only use 11 million of them for pre-training.

Pre-training data for web texts. Following MarkupLM (Li et al., 2022a), we take advantage of the large-scale dataset Common Crawl[3], which contains petabytes of web pages in raw formats. Specifically, text contents and HTML tags are both available for each web page. According to (Li et al., 2022a), the authors first filtered Common Crawl with fastText (Bojanowski et al., 2017) to remove non-English pages, then only kept common tags for saving disk storage, resulting in 24 million web pages for pre-training.

Specifically, we do not use any data augmentation or ensemble strategies for pre-training. We leverage AdamW optimizer (Loshchilov and Hutter, 2019) with learning rate 5e-5 and epsilon 1e-8. Moreover, we linearly warm up in the first 5% steps. Experiments are conducted with 32 NVIDIA Tesla

[1] https://www.wikipedia.org/
[2] https://github.com/tesseract-ocr/tesseract
[3] https://commoncrawl.org/

#	Model	Pre-train			Downstream Tasks							
		P	D	W	MNLI-m	QNLI	SST-2	MRPC	SQuAD1.1 / 2.0	FUNSD	DocVQA	WebSRC
1	RoBERTa	✓			**87.6**	92.8	94.8	90.2	**92.2*** / 83.4*	-	-	-
2	LayoutLM		✓		-	-	-	-	-	79.3	69.2	-
3	MarkupLM			✓	-	-	-	-	-	-	-	74.5
4	XDoc_{100K}	✓			87.0	**93.0**	95.2	90.1	91.9 / 83.4	70.1	64.5	58.5
5	XDoc_{100K}		✓		86.7	91.3	94.5	89.9	91.4 / 82.9	87.3	69.4	58.6
6	XDoc_{100K}			✓	86.5	92.0	94.6	90.1	91.4 / 83.1	71.6	63.6	64.8
7	XDoc_{100K}	✓	✓		87.2	92.7	94.9	90.2	91.9 / **83.5**	85.7	69.1	57.5
8	XDoc_{100K}		✓	✓	86.4	91.6	**95.3**	91.0	91.7 / **83.5**	85.7	69.5	65.0
9	XDoc_{100K}	✓		✓	86.8	92.3	95.1	90.6	91.6 / 83.0	70.0	64.7	64.8
10	XDoc_{100K}	✓	✓	✓	86.2	92.8	95.2	**91.3**	91.7 / 83.0	86.4	68.3	67.0
11	XDoc_{500K}	✓	✓	✓	86.6	92.2	95.2	89.9	91.7 / 83.1	89.1	72.6	73.3
12	XDoc_{1M}	✓	✓	✓	86.8	92.3	**95.3**	91.1	92.0 / **83.5**	**89.4**	**72.7**	**74.8**

Table 1: Results on downstream tasks for various document formats. P, D, and W denote whether XDoc is pre-trained with plain, document, and web texts, respectively. Compared with methods designed for a specific format (#1∼#3), XDoc achieves comparable or even better performance. **Accuracy** is used for MNLI-m, QNLI, and SST-2 for evaluation. **F1 score** is used for MRPC, SQuAD, FUNSD, and WebSRC. **ANLS** is used for DocVQA. Digits marked with * denote that we re-implement the results since the original paper did not report them.

V100 GPUs with 32GB memory. For those experiments pre-trained for 100K steps, we set the batch size to 128, while using all plain text datasets, the subset of document text (1 million), and web text (1 million) datasets for pre-training. Besides, we set the batch size to 512 and leverage all datasets for experiments pre-trained for 500K and 1M steps. FP16 is used during pre-training for accelerating and saving GPU memory. Within each batch, we equally sample documents in different formats for pre-training (see more discussions in Appendix B).

3.3 Fine-tuning on Downstream Tasks

In this subsection, we utilize a wide range of downstream datasets to validate the ability of pre-trained XDoc in different formats. Specifically, for the plain texts, we leverage the widely-used GLUE benchmark (Wang et al., 2019) and SQuAD (Rajpurkar et al., 2016, 2018). For document texts, we use the form understanding dataset FUNSD (Jaume et al., 2019) and question-answering dataset DocVQA (Mathew et al., 2021). For web texts, we utilize the question-answering dataset WebSRC (Chen et al., 2021). In the following, we will first introduce the downstream datasets in each format, then demonstrate the experimental results in detail.

3.3.1 Fine-tuning on Tasks for Plain texts

Fine-tuning on GLUE benchmark We evaluate XDoc on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019), which contains 9 datasets in total for evaluating natural language understanding systems. Specifically, 4 datasets four of them, including MNLI-m, QNLI, SST-2, and MRPC, are used for evaluation. We fine-tune XDoc for 10 epochs with a learning rate of 2e-5 and a batch size 16. The linear warmup is used for the first 100 steps. We utilize accuracy as the evaluation metric for MNLI-m, QNLI, SST-2, and F1 score for MRPC.

The experimental results are shown in Table 1 and we leverage $\text{RoBERTa}_{\text{BASE}}$ (Liu et al., 2019) as the baseline (#1). According to #4, we notice that after pre-training with plain texts, the performance of XDoc is almost consistent with the baseline. It is intuitive since XDoc is initialized with $\text{RoBERTa}_{\text{BASE}}$ and the continued training will not affect the performance. Interestingly, we notice that if XDoc is pre-trained without plain texts (refer to #5, #6, and #8), the performance is still on par with the baseline, indicating that the knowledge of plain texts will not be easily forgotten when XDoc is pre-trained using other formats.

Fine-tuning on SQUAD V1.1 and V2.0 We further employ the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016, 2018) for evaluation. SQuAD contains two versions: SQuAD V1.1 and SQuAD V2.0. For V1.1, given a question, the answer can always be retrieved in the paragraph. By contrast, for V2.0, there are some questions that can not be answered, which is more challenging compared with V1.1. Specifically, XDoc is fine-tuned with 2 epochs for V1.1 and 4 epochs for V2.0.

We set the batch size to 16 and the learning rate to 3e-5. We use the F1 score as the evaluation metric.

We also utilize $\text{RoBERTa}_{\text{BASE}}$ (Liu et al., 2019) as the baseline (#1). As is demonstrated in Table 1, we notice that the performance does not fluctuate much under various pre-training settings (#4∼#12). Similar to the experiment results on the GLUE benchmark, XDoc is capable of achieving comparable performance when pre-trained in all formats (refer to #10∼#12).

3.3.2 Fine-tuning on Task for Document texts

Fine-tuning on FUNSD We utilize the receipt understanding dataset FUNSD (Jaume et al., 2019) to verify the ability of XDoc. Deriving from the RVL-CDIP dataset (Harley et al., 2015), FUNSD contains 199 noisy scanned documents (149 samples for training and 50 for test) with 9,709 semantic entities and 31,485 words. Specifically, we focus on the entity labeling task, i.e. labeling "question", "answer", "header", or "other" in the given receipt. Concretely, we fine-tune XDoc for 1000 steps with the a batch size 64 and a learning rate 5e-5. We utilize linear warmup for the first 100 steps. The coordinates are normalized by the size of images following (Xu et al., 2020). F1 score is adopted as the evaluation metric.

For a fair comparison, we choose $\text{LayoutLM}_{\text{BASE}}$ (#2) (Xu et al., 2020) as the baseline, which exploits the layout and text knowledge for tackling visually rich document understanding. Through the experimental results, we observe that XDoc can outperform the baseline by a large margin if document texts are used during pre-training. According to #10, the performance can be boosted by 7.1% if all formats are used for pre-training. Besides, we notice that the performance can be boosted further when XDoc is trained for more steps (further increase by 3.0% according to #12). In contrast, it is observed that the performance will heavily deteriorate if the document texts are absent during pre-training (decrease by 9.3% according to #9).

Fine-tuning on DocVQA For further validating the ability of XDoc on document texts, we utilize the document question-answering dataset DocVQA (Mathew et al., 2021), which contains 10,194/1,286/1,287 images with 39,463/5,349/5,188 questions for training/validation/test sets, respectively. We follow LayoutLMv2 (Xu et al., 2021a) to employ Microsoft Read API to produce OCR results and find the given answers heuristically. We evaluate XDoc on the evaluation set and the final scores are obtained by submitting the results to the official website[4]. We fine-tune XDoc for 10 epochs with a batch size 16 and a learning rate 2e-5. The linear warmup strategy is used for the first 10% steps. Following (Xu et al., 2020), we normalize the coordinates by the size of images. We use Average Normalized Levenshtein Similarity (ANLS) as the evaluation metric.

As $\text{LayoutLM}_{\text{BASE}}$ (Xu et al., 2020) did not report the results on DocVQA, we borrow the ANLS score from LayoutLMv2 (Xu et al., 2021a). Similar to the experimental results on FUNSD, we observe that the performance of XDoc highly depends on the participation of document texts during pre-training. For example, if XDoc is pre-trained without document texts, the performance drops by 4.7%, 5.6%, and 4.5% according to #4, #6, and #9. When pre-training with 100K steps using all formats, XDoc obtains comparable performance (refer to #10). Furthermore, XDoc outperforms the baseline when training with more training steps (refer to #11 and #12).

3.3.3 Fine-tuning on Task for Web Texts

Fine-tuning on WebSRC We employ the Web-based Structural Reading Comprehension dataset (WebSRC) (Chen et al., 2021) to verify the ability of XDoc on web texts. It contains 440K question-answer pairs collected from 6.5K web pages. The HTML source code, screenshots, and metadata are available in this dataset. The training/validation/test parts consist of 307,315/52,826/40,357 question-answer pairs. The answer is either a text span in the given web page or yes/no. We fine-tune XDoc for 5 epochs with a batch size 16, a learning rate 5e-5, and a linear warmup rate 0.1. F1 score is used as the metric.

We use $\text{MarkupLM}_{\text{BASE}}$ (Li et al., 2022a) as the baseline (#3). When XDoc is only pre-trained for 100K steps, we notice that the performance is subpar compared with the baseline. It is intuitive since MarkupLM is pre-trained with **three** pretext tasks, including masked language modeling, node relation prediction, and title-page matching. Interestingly, we observe that when training for more steps (#12), the performance of XDoc surpasses the baseline. Similarly, it is observed that the per-

[4]https://rrc.cvc.uab.es/?ch=17

Init	MNLI-m	FUNSD	WebSRC	Avg
Scratch	75.4	78.8	29.2	61.1
RoBERTa	**86.2**	**86.4**	**57.5**	**76.7**

Table 2: Results on the initialization of XDoc.

Layers	MNLI-m	FUNSD	WebSRC	Avg
0	86.4	85.0	54.7	75.4
1	86.2	**86.4**	**57.5**	**76.7**
2	**86.7**	84.8	55.0	75.5
3	86.4	86.1	55.7	76.1
$1^{\dagger}$	86.4	84.8	57.3	76.2

Table 3: Results on the symmetry and number of adaptive layers. † means that the document and web branches share the same adaptive layers.

formance will drop heavily if web texts are absent during pre-training (refer to #4, #5, and #7).

3.4 Discussions

In this subsection, we conduct experiments to validate the effectiveness of the components or training strategies in XDoc. Unless specified otherwise, all experiments are pre-trained with 3M data (1M for each format) for 100K steps. Moreover, we discuss the parameter and time efficiency.

The initialization of XDoc We try to randomly initialize the parameters of XDoc with normal distribution and the results are demonstrated in Table 2. We observe that XDoc trained from scratch performs worse on downstream tasks, e.g. the performance drops by 10.8% for MNLI-m, 7.6% for FUNSD, and 28.3% for WebSRC. Therefore, we choose to initialize XDoc with $\mathrm{RoBERTa}_{\mathrm{BASE}}$ for better pre-training.

The symmetry and number of adaptive layers We utilize adaptive layers, which are implemented by a sequence of Linear and ReLU layers, to enhance the representations of different parts such as the 2D position and XPath embedding. Here we attempt to explore the symmetry and the number of adaptive layers. In detail, "symmetry" means the document and web branches share the same adaptive layers. Additionally, we denote the number of layers as the number of ReLU layers (e.g. Layers=2 means *Linear-ReLU-Linear-ReLU-Linear* and Layers=0 means no adaptive layers are used). As is demonstrated in Table 3, we notice that the average performance reaches the best if only one adaptive layer is used. Moreover, if we apply different adaptive layers to the document and web branches, the average performance can be boosted by 0.5% compared with the counterpart (76.2%).

Parameter efficiency We demonstrate some analysis of parameters in Table 4. We observe that the word embedding and Transformer layers contain most of the parameters (124M), e.g. occupy 96.9%, 94.7%, and 89.2% of all the parameters for $\mathrm{RoBERTa}_{\mathrm{BASE}}$, $\mathrm{LayoutLM}_{\mathrm{BASE}}$, and $\mathrm{MarkupLM}_{\mathrm{BASE}}$, respectively. By sharing the word embedding, 1D position embedding, and Transformer layers across multiple text formats, the proposed XDoc is efficient in terms of parameter usage. In detail, the total amount of parameters is 398M for three single models, while XDoc only contains 146M parameters (146M/398M≈36.7%) but can be used for downstream tasks in multiple formats. Besides, the newly introduced adaptive layers only contain 4M parameters, which is almost negligible for the whole model (2.7%).

Time efficiency Apart from the newly introduced adaptive layer, the architecture of XDoc is similar to those models targeting one specific document format. Since the adaptive layer is lightweight, it will not take much time overhead. For example, when conducting inference on the DocVQA dataset, it costs 45 ms for a batch while the adaptive layer only consumes negligible 0.8 ms (1.8%). Hence, XDoc is efficient in terms of the time cost.

4 Related Work

In this section, we review the pre-trained methods for document understanding, ranging from plain, document, and web texts, respectively.

Pre-trained methods for plain texts The understanding of plain texts through pre-training has been extensively studied during the last decade (Devlin et al., 2018; Yang et al., 2019; Bao et al., 2020; Liu et al., 2019; Lewis et al., 2020; Lan et al., 2019; Jiang et al., 2020; He et al., 2021; Dong et al., 2019; Lample and Conneau, 2019; Lin et al., 2021; Xue et al., 2020). For example, GPT (Radford et al., 2019; Brown et al., 2020) utilizes Transformer (Vaswani et al., 2017) to conduct single-director masked-word prediction in an unsupervised manner. Besides, BERT (Devlin et al., 2018) utilizes two self-supervised tasks, including mask language modeling and next sentence

Methods	Word 39M	1D Position 4M	Transformer 85M	2D Position 3M	XPath 11M	Adaptive 4M	**Total**
RoBERTa	✓	✓	✓	-	-	-	128M
LayoutLM	✓	✓	✓	✓	-	-	131M
MarkupLM	✓	✓	✓	-	✓	-	139M
XDoc	✓	✓	✓	✓	✓	✓	146M

Table 4: Analysis of the parameter efficiency. XDoc shares most parameters across different formats, including word embedding, 1D position embedding, and Transformer layers. We omit some layers that contain negligible parameters such as segment embedding layers and LayerNorm layers. All the comparison models are in **base** size.

prediction to obtain the robust representations of words based on Transformer. SpanBERT (Joshi et al., 2020) and ERNIE (Zhang et al., 2019) try to mask consecutive text spans so as to construct a more challenging pre-train task. In (Dong et al., 2019), the authors used different kinds of attention masks to enable one-direction and bi-direction attending. XLNet (Yang et al., 2019) introduces generalized autoregressive pre-training framework that utilizes a permutation language modeling objective. ELECTRA (Clark et al., 2020) first samples some candidates for the masked words and then uses a discriminator to predict whether a given token is replaced.

Pre-trained methods for document texts Benefiting from the public large-scale document dataset (Lewis et al., 2006), pre-training has become the de facto standard for analyzing document texts (Zhang et al., 2020; Wang et al., 2021; Xu et al., 2021b; Li et al., 2022b; Appalaraju et al., 2021; Garncarek et al., 2021; Gu et al., 2022b,a; Wu et al., 2021; Wang et al., 2022a). LayoutLM (Xu et al., 2020) makes the first attempt to combine the Layout knowledge during pre-training to obtain robust contextual features for document texts. Based on LayoutLM, LayoutXLM (Xu et al., 2021b) utilizes multilingual document text datasets for pre-training. StructuralLM (Li et al., 2021) jointly utilizes cell and layout information from scanned documents to make the representations more robust. LayoutLMv2 (Xu et al., 2021a) introduces a multi-modal architecture by combining additional image tokens in the Transformer. BROS (Hong et al., 2022) utilizes the token-masking and area-masking strategies for tackling information extraction tasks. XYLayoutLM (Gu et al., 2022b) proposes an Augmented XY-Cut algorithm to exploit proper reading orders during pre-training. Recently, LayoutLMv3 (Huang et al., 2022) pre-trains the text branch and image branch simultaneously using Mask Language Modeling and Mask Image Modeling tasks, which makes it a robust model for tackling text-centric and image-centric tasks.

Pre-trained methods for web texts Compared with plain and document text analysis, the understanding of web texts is less studied and is more challenging since the layout of each website is not fixed (i.e. depending on the resolution of devices). MarkupLM (Li et al., 2022a) takes the first attempt to incorporate web-based knowledge during pre-training while utilitzing three pretext tasks, including masked language modeling, node relation prediction, and title-page matching. Further, based on MarkupLM, DoM-LM (Deng et al., 2022) introduces a new pre-training task predicting masked HTML node. WebFormer (Wang et al., 2022b) simultaneously feeds text features and image features to the multi-modal Transformer while constructing rich attention patterns between these tokens.

Generally, although the mentioned methods show impressive performance in one specific format, they can not be transferred to tackle other formats. To mitigate this problem, the proposed XDoc is a scalable and flexible framework that is friendly to a wide range of formats, thus bringing much convenience for people.

5 Conclusion and Future Work

In this paper, we propose XDoc, a unified framework that can tackle multiple document formats (e.g. plain, document, and web texts) in one model. For parameter efficiency, XDoc shares most parameters, including the word embedding, 1D position embedding, and Transformer layers, across different document formats. The experimental results show that with only 36.7% parameters, XDoc can achieve comparable or even better performance on downstream tasks spanning various document

formats. For future work, we will consider exploiting the image features during pre-training to tackle image-centric tasks and designing more unified pre-training tasks for various document formats.

Limitations

As XDoc only leverages the text and layout information for pre-training, it is not suitable to tackle some image-centric tasks such as page object detection. Besides, XDoc only uses masked language modeling as the only pre-training task in this version. For future work, we will consider designing more unified pre-training tasks for various document formats.

References

Vanya Cohen Aaron Gokaslan. 2019. Openwebtext corpus. http://web.archive.org/save/http://Skylion007.github.io/OpenWebTextCorpus.

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

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *ICML*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*.

Lu Chen, Xingyu Chen, Zihan Zhao, Danyang Zhang, Jiabao Ji, Ao Luo, Yuxuan Xiong, and Kai Yu. 2021. Websrc: A dataset for web-based structural reading comprehension. In *EMNLP*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Lei Cui, Yiheng Xu, Tengchao Lv, and Furu Wei. 2021. Document ai: Benchmarks, models and applications. In *CCL*.

Xiang Deng, Prashant Shiralkar, Colin Lockard, Binxuan Huang, and Huan Sun. 2022. Dom-lm: Learning generalizable representations for html documents. *arXiv preprint arXiv:2201.10608*.

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

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *NeurIPS*.

Łukasz Garncarek, Rafał Powalski, Tomasz Stanisławek, Bartosz Topolski, Piotr Halama, Michał Turski, and Filip Graliński. 2021. Lambert: Layout-aware language modeling for information extraction. In *ICDAR*.

Jiuxiang Gu, Jason Kuen, Vlad I Morariu, Handong Zhao, Nikolaos Barmpalios, Rajiv Jain, Ani Nenkova, and Tong Sun. 2022a. Unified pretraining framework for document understanding. *arXiv preprint arXiv:2204.10939*.

Zhangxuan Gu, Changhua Meng, Ke Wang, Jun Lan, Weiqiang Wang, Ming Gu, and Liqing Zhang. 2022b. Xylayoutlm: Towards layout-aware multimodal networks for visually-rich document understanding. In *CVPR*.

Adam W Harley, Alex Ufkes, and Konstantinos G Derpanis. 2015. Evaluation of deep convolutional nets for document image classification and retrieval. In *ICDAR*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *ICLR*.

Teakgyu Hong, Donghyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. 2022. Bros: A pre-trained language model focusing on text and layout for better key information extraction from documents. In *AAAI*.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. *arXiv preprint arXiv:2204.08387*.

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

Zi-Hang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. Convbert: Improving bert with span-based dynamic convolution. In *NeurIPS*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

David Lewis, Gady Agam, Shlomo Argamon, Ophir Frieder, David Grossman, and Jefferson Heard. 2006. Building a test collection for complex document information processing. In *SIGIR*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

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

Junlong Li, Yiheng Xu, Lei Cui, and Furu Wei. 2022a. Markuplm: Pre-training of text and markup language for visually-rich document understanding. In *ACL*.

Junlong Li, Yiheng Xu, Tengchao Lv, Lei Cui, Cha Zhang, and Furu Wei. 2022b. Dit: Self-supervised pre-training for document image transformer. *arXiv preprint arXiv:2203.02378*.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, et al. 2021. Few-shot learning with multilingual language models. *arXiv preprint arXiv:2112.10668*.

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*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *WACV*.

Sebastian Nagel. 2016. Cc-news. http://web.archive.org/save/http://commoncrawl.org/2016/10/news-dataset-available.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

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 *NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Jiapeng Wang, Lianwen Jin, and Kai Ding. 2022a. Lilt: A simple yet effective language-independent layout transformer for structured document understanding. In *ACL*.

Qifan Wang, Yi Fang, Anirudh Ravula, Fuli Feng, Xiaojun Quan, and Dongfang Liu. 2022b. Webformer: The web-page transformer for structure information extraction. In *WWW*.

Zilong Wang, Yiheng Xu, Lei Cui, Jingbo Shang, and Furu Wei. 2021. Layoutreader: Pre-training of text and layout for reading order detection. In *EMNLP*.

Te-Lin Wu, Cheng Li, Mingyang Zhang, Tao Chen, Spurthi Amba Hombaiah, and Michael Bendersky. 2021. Lampret: Layout-aware multimodal pretraining for document understanding. *arXiv preprint arXiv:2104.08405*.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. 2021a. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. In *ACL*.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *KDD*.

Yiheng Xu, Tengchao Lv, Lei Cui, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, and Furu Wei. 2021b. Layoutxlm: Multimodal pre-training for multilingual visually-rich document understanding. *arXiv preprint arXiv:2104.08836*.

Yiheng Xu, Tengchao Lv, Lei Cui, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, and Furu Wei. 2022. Xfund: A benchmark dataset for multilingual visually rich form understanding. In *ACL*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

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 MM*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *ACL*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*.

A Details of XPath embedding

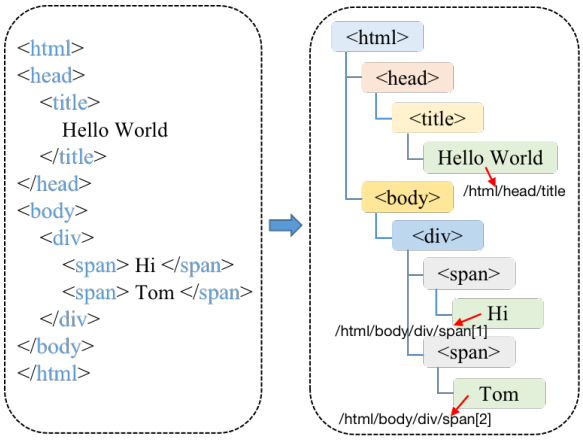
As is illustrated in Figure 4, each web page can be represented as a DOM (Document Object Model) tree based on the corresponding HTML source code. In addition, XPath is a query language for selecting nodes based on the DOM tree. For example, the XPath of the text "Tom" can be represented as "/html/body/div/span[2]", where the texts denote the order of tag name traversed from the root node and the subscripts stand for the index of a node when more than one nodes have the same tag name under a parent node. For those tags without subscripts, we simply set the subscripts to 0. Following MarkupLM, we filter some unimportant tags and only reserve some common tags such as <html>, <body>, , , <li>, <a>, etc.

To construct the XPath embedding for a given subword s_i^W, we first denote its XPath as $xpath_i^W = [(tag_1, sub_1), (tag_2, sub_2), ..., (tag_D, sub_D)]$, where D means the maximum depth of the sequence, while tag_j and sub_j denotes the tag name and subscript at the j-th depth, respectively. For example, we represent the XPath of the text "Tom" as $[(html, 0), (body, 0), (div, 0), (span, 2)]$. Subsequently, for each pair (tag_j, sub_j) at depth j, we calculate its embedding ts_j by adding up the tag embedding and subscript embedding:

$$ts_j = \text{TagEmb}_\text{j}(tag_j) + \text{SubEmb}_\text{j}(sub_j) \quad (5)$$

Please note that the embedding layer of tags and subscripts vary across different depths. Finally, we concatenate the embedding of all pairs to construct the XPath embedding:

$$\text{XPathEmb}(xpath_i^W) = [ts_1; ts_2; ...; ts_D] \quad (6)$$



(a) HTML source code (b) DOM tree and XPath

Figure 4: Illustrations of the way to construct XPath based on the corresponding HTML source code. Some examples of XPath are indicated using red arrows.

B Balance of Pre-training Data

We attempt to use different sampling ratios for different formats during pre-training and the experimental results are shown in Table 5. For example, "3:1:1" denotes that there are approximately 60% plain texts, 20% document texts, and 20% web texts in a batch. We notice that the average performance reaches the best (76.7%) if we use the balanced sampling strategy. Interestingly, we observe that the sampling ratio with respect to one specific format does not positively correlate with the performance. For instance, when "P:D:W" is set to 1:1:3, the performance on WebSRC is the worst (55.4%) among all experiments.

P:D:W	MNLI-m	FUNSD	WebSRC	Avg
1:1:1	86.2	**86.4**	**57.5**	**76.7**
3:1:1	86.7	83.8	56.7	75.7
1:3:1	86.7	84.8	56.6	76.0
1:1:3	**87.1**	83.7	55.4	75.4

Table 5: Results on the balance of pre-training datasets. P:D:W denotes the ratio of plain, document, and web texts in a batch, respectively.