Pixel-Base Auto-Regressive Language Modeling

Xiyang Liao

Master of Science

Data Science

School of Informatics

University of Edinburgh

2023

Abstract

Pixel-based language models are inspiring researchers of a new approach to represent natural languages, and it is shown to perform similarly compared to subword-based language models. By rendering the text as images, we could get rid of the previous restrictions of model vocabulary and provide a more robust and generalizable embedding channel of language scripts. However, it is under exploration how to generate new content with pixel-based language models and what is the better configuration of the rendered images. This paper introduces PIXAR, a pixel-based auto-regressive language model. By rendering the text as images with binary pixel values, PIXAR is trained to generate image patches auto-regressively in the decoder, based on the corrupted encoder output. Pretrained on the same training set as PIXEL, PIXAR is evaluate on predominantly English downstream tasks covering both token-level and sentence-level understanding. We find that PIXAR outperform PIXEL in token-level understanding tasks and similarly on sentence-level tasks at early pre-training steps. Moreover, after trying different combination of pre-training settings from architecture, masking and input representations, we find that the binary pixel values provide more robust performance in task scores compared to grey-style pixel values. And PIXAR's encoder-decoder architecture with span masking reach consistently strong performance among all the settings.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Xiyang Liao*)

Acknowledgements

This thesis is the culmination of a long and arduous journey-one that I have only been able to make as a results of the dedicated support I have received from so many people along the way.

The deepest and sincerest gratitude goes to my supervisor Antonio Vergari and associate supervisor Suglia, Alessandro for their continuous and invaluable guidance throughout my research. It is a great honor and privilege that I was given the opportunity to under their supervision. I would like to thank them for the patience, support, empathy and great sense of humor. The thoughts and comments I got from my supervisors are so helpful in directing me to continue my research. I addition, I'm also thankful to my partner Yintao Tai, who gave me great support in sharing me with his experiences in conducting the experiments. I was inspired and impressed by his opinions and advice greatly and it saved me a lot of work with his support. I really appreciate that.

Sincerely wishes to your, may everything be pleasant and pleasant to you.

Table of Contents

Chapter 1

Introduction

The following section is modified from my IPP report.

Self-supervised representation learning has become a popular technique for enhancing the state-of-the-art performance in natural language processing tasks. In recent years, there has been a tremendous increase in the size of large language models (LLMs), with billions of parameters integrated into their architectures. Correspondingly, pre-training these models requires millions or even billions of rows of data, which is often not available through labeled data alone. By learning to reconstruct itself, self-supervised representation learning enables models to leverage large unlabeled datasets, such as C4 (Raffel et al., 2020) and The Pile (Gao et al., 2021), contributing significantly to the success of pre-trained language models (PLMs) like BERT (Devlin et al., 2019), GPTs (Brown et al., 2020; OpenAI, 2023) and XLM-R (Conneau et al., 2020).

However, these language models operate on a finite set of inputs, such as words, sub-words, characters or bytes. This results in a vocabulary bottleneck when attempting to input units that are not included in the model's vocabulary. This problem is exacerbated as the number of supported languages increases and the challenge of dealing with noisy inputs becomes more complex. Inevitably, issues such as out-of-vocabulary words and memory limitations in the embedding layer and output softmax layer arise when implementing a finite vocabulary. As a result, a smaller and more generalized vocabulary must be constructed to address this issue. While character or byte-based vocabularies are smaller, they can lead to increased sequence length (Keren et al., 2022), which will reduce the speed and efficiency for both training and inference.

To tackle the vocabulary bottleneck, current language models use sub-word units to avoid the problem of extremely large vocabulary size, maintain an acceptable sequence length, and support open vocabulary processing. However, while practical in a monolingual context, sub-word segmentation poses challenges in a multilingual context. It is computationally prohibitive to fully represent the vocabulary of each language, and a mismatch between the vocabularies in pre-training and fine-tuning may lead to sub-optimal segmentation, degrading performance in domain transfer. Consequently, this results in degraded cross-lingual performance for languages that are underrepresented in the training data.

A proposed solution to address the vocabulary bottleneck is to rethink language modeling as a visual recognition task, given the orthographic similarities between characters across language scripts and the potential for visually representing the meaning of language in writing systems (Rust et al., 2022). This idea builds on the work of Salesky et al. (2021), who trained a machine translation model with ”visual text representations” in the encoder instead of sub-words. Rust et al. (2022) developed a pixel-based encoder of language model (PIXEL) based on the masked auto-encoding visual Transformer (ViT-MAE) (He et al., 2021) to reconstruct pixels in masked image patches. PIXEL solves the vocabulary bottleneck by rendering text as a sequence of fixed-sized patches, which are then processed by a Vision Transformer encoder (Dosovitskiy et al., 2020) to get their latent embeddings. This approach also prevents the cost of longer sequences. Compared to BERT with the same pre-training configuration, PIXEL achieves comparable performance in many syntactic and semantic tasks and demonstrates excellent robustness when transferring to unseen scripts.

However, similar to BERT, the bidirectional architecture of PIXEL prevent it from doing generative tasks. Moreover, PIXEL renders the text in grey style, which is a 0 to 1 continuous space. This is contradictory to the idea of discrete word embeddings used by sub-word based language models. Therefore, we propose to investigate the following two research questions:

1. Can we build a pixel-based language model with generation ability?
2. Can we simplify the way to represent the texts with images?

In this paper, we present PIXAR, a **PIX**el-based **A**uto-**R**egressive language model. PIXAR is a sequence-to-sequence denoising auto-encoder that can be adapted to various downstream tasks. With a similar architecture to transformer-basd neural machine translation models, PIXAR is pre-trained by reconstruct the whole image sequence in the decoder conditioned on the corrupted image in the encoder. We rethink the pixel-based generation as image generation and adapt the success of ImageGPT(Chen et al., 2020) with a adjustment to patch-wise instead of pixel-wise auto-regressive generation. This prevent the extreme long sequence length and better preserve the positional information for consecutive image patches. Predicted patch embeddings can be used for downstream tasks at ease. Moreover, we render the text with binary pixel values with only one channel, which is a much simplified and can provide discrete embeddings for different image patches. PIXAR can be seen as combination of ViT (the bidirectional encoder) and imageGPT (the auto-regressive decoder) as shown in Figure 3.3.

PIXAR is pre-trained on the same corpus as PIXEL. We evaluate PIXAR on a variety of token-level and sentence-level understanding tasks. Benchmarked with the intermediate checkpoints of PIXEL, the results show that PIXAR performs better in token-level understanding and comparable in sentence-level understanding tasks at early pre-training steps. Also to explore the effect of different pre-training configuration, we conduct ablation analysis to study how the architecture, masking and input representation will impact PIXAR's performance on the downstream tasks. We find that the model with encoder-decoder architecture and binary pixel values provides consistently strong and stable performance compared to other configurations.

Chapter 2

Literature Review

The following section is taken from my IPP report.

2.0.1 Denoising Auto-encoding Transformers

Auto-encoding neural networks are a type of self-supervised feature extraction technique used to compress input vectors into a lower-dimensional space in the encoder, followed by reconstruction of the original vectors in the decoder. After sufficient pre-training steps, the hidden states from the encoder can be considered as a dense representation of the input vectors and can be combined with task-specific neural networks to fine-tune the model for downstream tasks. To increase the robustness of the model, various types of noise, such as dropout, masking or swapping words, are often added to corrupt the input vectors. By learning to reconstruct the uncorrupted vectors, the models can improve their understanding of the unseen vectors. This idea is reflected in denoising auto-encoders (DAEs)(Vincent et al., 2008). In natural language processing, auto-encoding architecture is widely used as pre-trained language models for word embedding, which uses a dense vector to capture the semantic meaning of each word in a fixed vocabulary. The auto-encoding pre-training technique has achieved great success after the introduction of BERT(Devlin et al., 2019), which is a bidirectional encoder representation from Transformers(Vaswani et al., 2023). BERT uses masked language modeling and next sentence prediction as pre-training formulation to construct a task-agnostic word embedding solution and improves the state-of-art performance for a range of natural language processing tasks. There are many derived models, such as Roberta(Liu et al., 2019), T5(Raffel et al., 2020), BART(Lewis et al., 2019), XLM(Lample and Conneau, 2019), and ALBERT(Lan et al., 2020), that demonstrate the great generalization ability of auto-encoding pre-training on downstream tasks.

Although originally developed for natural language processing, the Transformer architecture has also been applied to computer vision tasks(Dosovitskiy et al., 2020) through a supervised pre-training scheme. He et al. (2021) proposed to use a masked auto-encoder (MAE) architecture to learn pixel representations in a self-supervised manner using pixel reconstruction loss. This framework involves a combination of a Vision Transformer with a lightweight decoder and has been found to perform well on other modalities such as video(**?**Feichtenhofer et al., 2022), audio(Baade et al., 2022), and even in multi-modal settings(Geng et al., 2022).

2.0.2 Vocabulary in Pre-trained Language Models

Vocabulary construction directly determines the granularity of text units represented by the language models. Traditional models based on word-level representation and closed-vocabulary suffer from limitations in dealing with rare and novel words, which can result in the out-of-vocabulary problem. To address this issue, modern neural networks use sub-word segmentation techniques(Sennrich et al., 2016) to break down entire sentences into sub units and extend closed-vocabulary models to open-vocabulary models. This allows for more robust and flexible language modeling, as it can handle words that are not presented in the training data.

Sub-word segmentation strikes a balance between what can be represented in the embedding layers and the sequence length after segmentation. The breakthrough for sub-word tokenization was achieved with Byte-Pair-Encoding (BPE;(Sennrich et al., 2016)). Originally developed as a data compression technique(Gage, 1994), BPE is now used as a vocabulary construction technique by PLMs. When creating a new vocabulary, BPE replaces the most frequent adjacent character pair with a new symbol representing that pair. This process is repeated iteratively until all frequent occurrences of character pairs are identified. During testing, merging is executed by looking up all the recorded merges from the training. BPE has also been adapted by GPT-2(Radford et al., 2019), where it merges bytes instead of characters. A similar algorithm used by BERT with different merging rule on characters is 'WordPiece'(Schuster and Nakajima, 2012). WordPiece uses a per-word left-to-right longest-match-first strategy, computing scores for character pairs to prioritize the merging of pairs whose components are less frequent in the vocabulary. Following the idea of evaluate the sub-word segmentation by their performance in the language model, a simple uni-gram language model (UnigramLM) was proposed by Kudo (2018) to remove sub-word units with the lowest probability on every iteration from a starting inclusive sub-word vocabulary until the expected vocabulary size is reached.

In multilingual pre-trained language models such as mBERT(Devlin et al., 2019), XLM-R(Conneau et al., 2019), and mT5(Xue et al., 2020), the vocabulary bottleneck prevents us from fully representing the vocabulary of each individual language. Additionally, Rust et al. (2020) pointed out that BERT-based Transformers exhibit a bias towards high-resource languages, leading to degraded cross-lingual performance for underrepresented languages. Therefore, there is a pressing need to build ”tokenization-free” language models. While character-level models are robust to noise and out-of-vocabulary problems, they often result in longer sequence lengths. One alternative approach is to represent text using the byte sequences obtained from the UTF-8 encoding library, rather than relying on linguistic and statistical features. Although the byte code was developed to cover all characters in all writing systems, ByT5(Xue et al., 2022) demonstrated that byte-level models exhibit the same characteristics as character-level models in terms of the benefits of robustness and the drawbacks of longer sequence lengths.

Driven by the way human readers perceive and process text, a new approach has been proposed that utilizes visual text representation instead of byte codes to represent text. This approach involves creating embeddings through convolutions of the character images with shared components, which enables generalization to unseen characters (Wang et al., 2020; Meng et al., 2020; Sun et al., 2021). Recent work has introduced the concept of ”visual text representation” (Salesky et al., 2021; Mansimov et al., 2020), which involves rendering text into images and decomposing pixels for translation. This novel technique not only circumvents the limitations of character-level and byte-level models, but also performs competitively across writing systems by incorporating both topological and logo-graphic features. Pixel-based models exhibit greater robustness to noisy inputs.

2.0.3 Generative language models

Generative models learn to create new data based on their knowledge from the dataset. Unlike discriminative models that predict the conditional distribution $P(Y|X)$, generative models aim to compute the joint distribution $P(X,Y)$ of the target Y and observation X. This makes generative models naturally enjoy the abundant resources of unlabeled data used in unsupervised algorithms. Variational Auto-encoders (VAEs)(Kingma and Welling, 2022) are notable architectures for probabilistic generative models whose encoders map the input into a distribution over the latent space and then the model samples latent representations to reconstruct the input sequence in the decoder. Since the advent of Transformers(Vaswani et al., 2023), auto-regressive models, such as sequence-to-sequence models, have shown outstanding effectiveness in many natural language processing tasks. Original developed for neural machine translation, sequence-to-sequence model(Lewis et al., 2019; Raffel et al., 2020) is a branch of generative models that compresses the information of the input sequence in the encoder output and let the decoder to generate conditioned on it. The decoder of is simply a feed-forward network that uses past values to predict future values in a unidirectional manner (usually left-to-right). For example, BART(Lewis et al., 2019) leverages both the bidirectional representation of BERT and generative ability of GPT to achieve state-of-art performance on both discriminative tasks and generation tasks.

Generative Pre-trained Transformers (GPTs) are a set of large language models that have gained significant attention due to their remarkable text generation quality. These models employ the transformer architecture's ability to process sequential data for text generation and are pre-trained on large unlabelled datasets, allowing them to generate human-like text. Despite functioning as an auto-regressive decoder, the GPT-3 model, with 175 billion learnable parameters, can perform both natural language understanding and generation tasks with exceptional proficiency. ImageGPT(Chen et al., 2020) adapt the framework of GPT to do image generation by re-configuring the RGB style pixel values into one channel and let the GPT decoder to predict the pixel values at the raster order. The reconfiguration of pixel values is further refined by DALLE(Ramesh et al., 2021) to learn a discrete variational auto-encoder that can compress the 256×256 RGB images into 32×32 grid image tokens, each element of which can assume 8192 possible values. As a consequence, the context size of the model is reduced without a degradation in visual quality. Although we do not adapt their code book of mapping pixel values into a latent space with lower resolutions, their methods are still beneficial for further simplifying the input representations of rendered text.

Chapter 3

Approach

3.1 Text Rendering

To train PIXAR, we need to first render the texts to images as the input to the encoder and the target to the decoder during the pre-training. Following (Rust et al., 2022)'s work, we use their text renderer which can covert one or more pieces of text into an RGB images $x \in \mathbb{R}^{H \times W \times C}$. PIXEL's text renderer creates the images in a grey style, where the pixel values are in the continues space between 0 and 1. We build a channel to transform it into binary pixel values with only 0s and 1s, which can provides discrete embeddings for image tokens. We set the height $H = 8$, the width $W = 4232$ and the input channels $C = 1$. In this setting, the rendered image is equal to a sequence of 529 image patches of size 8 x 8 pixels, which is smaller than the 16 x 16 patch size of PIXEL. To fit the font on the 8 x 8 grid, we choose the PixeloidSans-mLxMm font type, which is designed over the 8 x 8 grid map. Similar to text tokenizers, the text renderer supports the common ultilizations that are required for natural language processing tasks, such as using black patches as end-of-sequence (EOS) markers and white patches as padding. The text with longer sequence than the maximum length are truncated.

3.2 Architecture

PIXAR is a denoising auto-encoder that maps a corrupted image to its source image. We adapt the architecture of sequence-to-sequence model with a bidirectional encoder over corrupted image sequence and a left-to-right auto-regressive decoder. The encoder is a 12-layer ViT encoder(Dosovitskiy et al., 2020) with 86M parameters and the decoder is a 12-layer GPT decoder(Radford et al., 2018) with 113M parameters. PIXAR differs

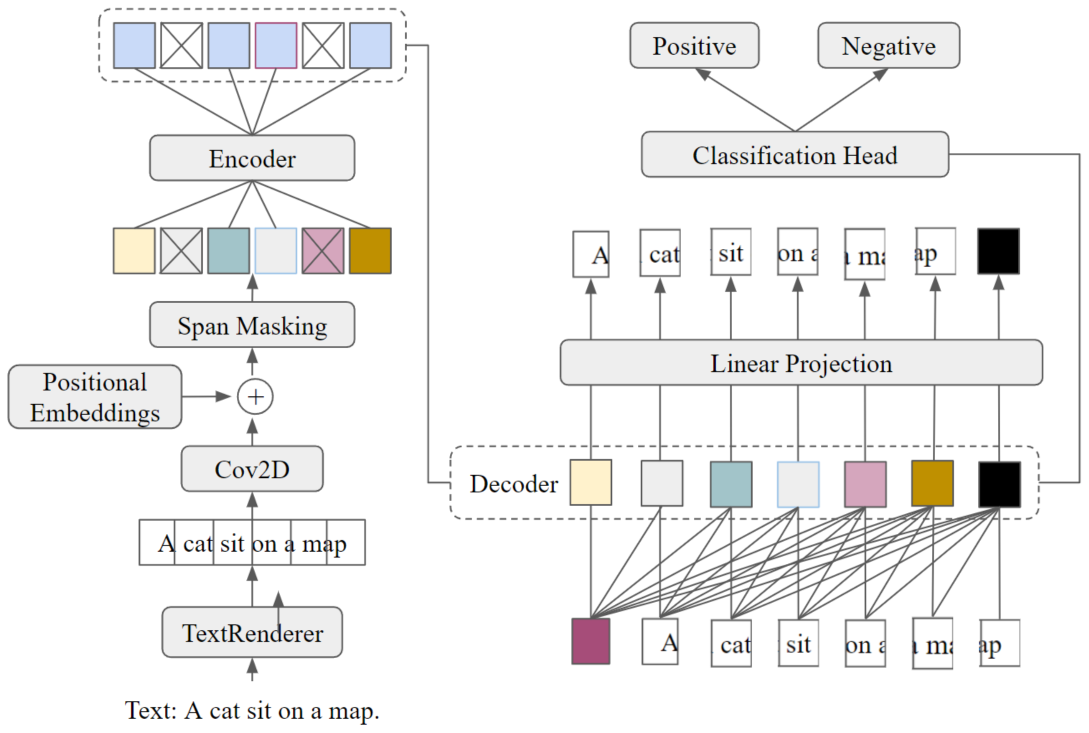


Figure 3.1. An overview of PIXAR's architecture. We use a sequence-to-sequence framework with a bidirectional encoder and a left-to-right auto-regressive decoder for pre-training. For finetuning, the last decoder hidden states is fed into the task-specific classification head.

with PIXEL in the following settings: (1) PIXAR use a auto-regressive generative decoder compared to the bidirectional lightweight decoder of PIXEL. Although both decoders are conditioned on the encoder output, generative decoder outputs predicted logits once a time without the knowledge of future logits while bidirectional decoder output predicted logits with the comprehensive knowledge of the previous and future logits; (2) PIXAR uses smaller patch size and channel size and binary pixel values when rendering the text; (3) PIXAR is trained on the binary classification loss on the whole sequence, while PIXEL is trained on the mean squared error on the masked image patches only. The overview of the PIXAR architecture is illustrated as Figure 3.3, with more details in Appendix A.1.

3.2.1 Span Masking

We follow the practice of PIXEL in using span masking to add noise into the encoder input. Proposed by T5(Raffel et al., 2020) and SpanBERT(Joshi et al., 2020), span masking is developed to mask contiguous random spans, rather than random tokens, to makes the masked units of text more meaningful and contain more information because full words or phrases are usually masked. Therefore, PIXAR is able to gain more abstractive inference ability. We also tried use random masking and the comparison of these two masking procedure is demonstrate in the ablation studies. Due to the lack of computational resources, we do not construct a comprehensive hyper-parameter search and just follow the configuration of PIXEL to use the masking ratio of 0.25 and maximum span length of 6.

3.2.2 Encoder

The bidirectional transformer encoder takes an input image with the shape of [num_channel, patch_size, seq_len×patch_size]. The input image is first fed into the embedding layer with a 2D convolution layer using patch_size as both kernel size and stride to produce one d-dimensional patch embeddings for each image patch. Following PIXEL(Rust et al., 2022), we add fixed sinusoidal positional embeddings to these patch embeddings. Then, the patch embeddings are fed into a stack of L blocks, with each block produces an intermediate embedding with the same dimension as patch embeddings. Given input tensor h^l, we use the ViT(Dosovitskiy et al., 2020) formulation of the transformer encoder block as follows:

$$n^l = \text{layer\_norm}(h^l)$$

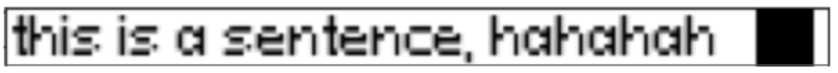
$$a^l = h^l + \text{multi-head-attention}(n^l)$$
$$h^{l+1} = a^l + \text{mlp}(\text{layer\_norm}(a^l))$$

Following PIXEL and ViT-MAE(He et al., 2021), PIXAR encoder only processes unmasked patches instead of including masked tokens during pre-training. This can not only lead to smaller memory usage used during training, but also can accelerate the training speed. Moreover, the mismatch between pre-training and fine-tuning is eliminated since mask tokens will not be inserted during fine-tuning. Unlike PIXEL and ViT-MAE, we do not add the special CLS embeddings to the beginning of the sequence and the encoder output is only used for the cross-attention computation in the decoder.

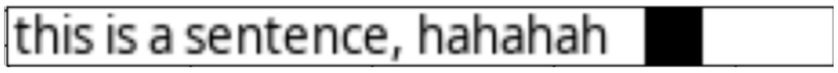
3.2.3 Decoder

The left-to-right decoder uses sequence-to-sequence architecture from (Vaswani et al., 2023). The decoder block formulation is similar to that is used in encoder with additional cross-attention over the encoder output. We first build a dummy head with the same dimension as decoder model hidden size. This is used during both training and inference as the start of sequence. The source image is patchified into the shape of [seq_len, num_channel×patch_size×patch_size] to get our label for training. To ensure that the prediction at next step is only conditioned on the previous steps, standard triangular casual mask is implemented to the $n \times n$ matrix of logits.

We made a adjustment from the pixel-wise to patch-wise auto-regressive generation, where the last decoder hidden state is linearly projected into the same size as the label. Instead of adding a softmax layer and get the predicted token index with the highest probability, we just keep the output with the size of [num_channel×patch_size×patch_size] as our prediction, which is then unpatchfied into the size of image patch [num_channel, patch_size, seq_len×patch_size] and then pass through the shared 2D convolution layer from the encoder to get the decoder input embedding for next step. Unlike in the encoder where the whole sequence is transformed into a sequence of patch embeddings at the same time, in the decoder the image patches is transformed into patch embeddings once at a time in a auto-regressive manner. PIXAR is trained with a binary classification loss between the prediction image patches and target. The loss is computed for the whole sequence except the padding blank image patches.



(1) PIXAR - PixeloidSans-mLxMm



(2) PIXEL - GoNotoCurrent

Figure 3.2. Comparison of the font type used by PIXAR and PIXEL. By setting the same patch size and font size we find that for the same number of image patches, PIXAR will cover smaller number of words.

3.3 Pre-training

Following PIXEL(Rust et al., 2022) and BERT(Devlin et al., 2019)'s pre-training setup, PIXAR is pre-trained on a rendered version of the English Wikipedia and the BookCorpusZhu et al. (2015). We use different configuration for text rendering compared with PIXEL. In total. PIXAR is pretrained for 0.1M steps with batch size 128. Our rendered Wikipedia are converted into 23.5M examples and the Bookcorpus has 6.7M examples. This is larger than PIXEL's rendered dataset, which has only 11.4M Wikipedia examples and 1.1M Bookcorpus examples. This is because that the adjusted font type will flatten the words and can only fit smaller number of tokens into the same number of image patches. The examples of rendered text with the font type used by PIXEL and PIXAR are shown in Figure 3.4. Therefore, for a fixed number of documents, the adjusted rendering configuaration will generate more samples. This also means that for the same number of training steps, our model might cover less word tokens compared the PIXEL. We select AdamW(Loshchilov and Hutter, 2019) as our optimizer, warming up the learning rate linearly over the first 5K steps, peaking at 1.5e-4, then decaying via a cosine function to a minimum learning rate of 1e-5. In average, training PIXAR takes 20 hours on 16×16 GB Tesla V100 GPUs.

The choice of pre-training objective is crucial since it offers the mechanism through which the model acquires generic knowledge to use in downstream tasks. PIXEL is a masked auto-encoder with the objective to reconstruct the masked image patches given the unmasked input. This objective was modified from (Devlin et al., 2019), which is designed for encoder-only models and the loss is computed only on the masked tokens during training. To make it suitable in our encoder-decoder setup, we make

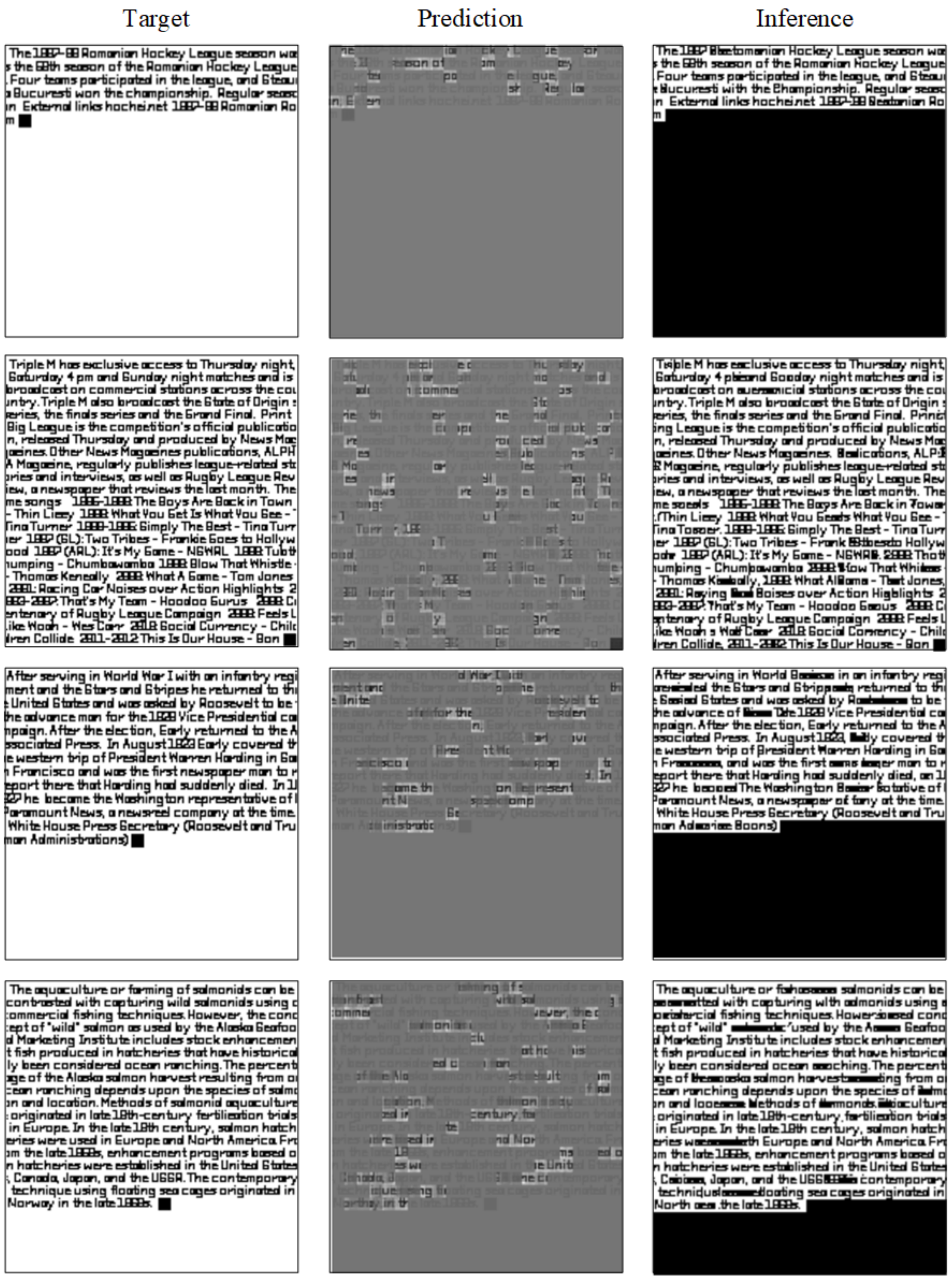


Figure 3.3. PIXAR image reconstruction with different masking types. The first two rows are random masking and the last two rows are span masking. The prediction column is the next-patch generation where given the previous golden image patch, we generate only the next image patch. The inference is generated from the start of sentence auto-regressively without seeing any golden image patch in the decoder. We apply a span masking with a ratio of 0.25. We find that PIXAR can learn to reconstruct image patches given either previous image patch or encoder hidden states of that specific position.

a modification in that we mask out 25 pre cent of the input image patches and the model is trained to reconstruct the whole uncorrupted image sequence. Similar masking objective was used by text-to-text models like MASS(Song et al., 2019) and T5(Raffel et al., 2020) as a simple variant of the BERT-style objective. The loss is computed over the whole sequence instead of the masked image patches only. This formulation of pre-training objective is proved to perform similarly but more efficiently compared with other variants of BERT-style objective.

Inspired from the auto-regressive pre-training objective of GPT(Radford et al., 2018) and imageGPT(Chen et al., 2020), our formulation of objective is demonstrated as follows: Given an dataset with the data as a sequence of image patches $x = (x_1, ..., x_n)$ where n is the maximum sequence length, each image patch x_i has a dimensionality of 64 (the total pixel amount that equals to the value of [num_channel$\times$patch_size$\times$patch_size]). The input sequence x is first fed into the encoder with a specified mask ratio to get the encoder hidden states h_e, then the decoder predict the density of each image patch $p(\hat{x}_i)$ auto-regressively conditioned on h_e as below:

$$p(\hat{x}_i) = p(\hat{x}_i | x_1, ..., x_{i-1}, \theta, h_e) \tag{3.1}$$

where θ indicates the parameters of the decoder. Here the model is trained to predict the next image patch $\hat{x}_i$ only given all the previous gold image patches ($x_1, ..., x_{i-1}$) in a teacher-forcing manner. For binary pixel values, we define the density of each pixel x_{ij} of the image patch x_i as:

$$p(\hat{x}_{ijt}) = \begin{cases} p(\hat{x}_{ij}), & if x_{ij} = 1 \\ \\ 1 - p(\hat{x}_{ij}), & otherwise \end{cases} \tag{3.2}$$

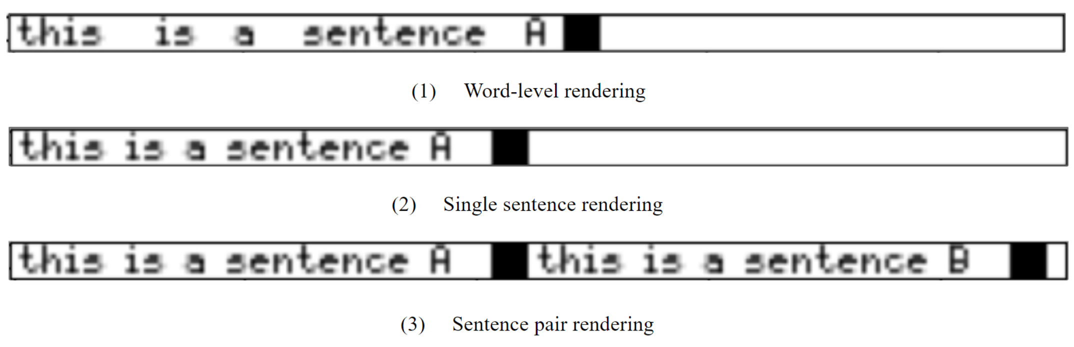
We train PIXAR by minimizing the focal loss(Lin et al., 2018) between the predictions and targets. Focal loss is used by many modern vision models(Kirillov et al., 2023; Cheng et al., 2021) for pre-training objectives and is proved to be more efficient than cross entropy loss.

$$Loss = \sum_{i=0}^{n} \sum_{j=0}^{64} -\alpha_t (1 - p(\hat{x}_{ijt}))^{\gamma} log(p(\hat{x}_{ijt})) \tag{3.3}$$

Focal loss is a improved version of cross entropy loss. It introduces a modulating factor $(1 - p(\hat{x}_i))^{\gamma}$ to assign larger importance to the less confident samples and make the model focus more on the harder samples. Also, α is a balancing variant to deal with class imbalance issues. γ is set to 2 following the best practice of Lin et al. (2018) and α is set to 0.3 for positive class according to the distribution of the binary values from the dataset.

3.4 Finetuning

PIXAR follows the practise of similar language models with encoder-decoder architectures to use the last hidden states of their decoders as the representations of the input sentences(Lewis et al., 2019; Raffel et al., 2020). Due to the restriction of computational resources, we do not construct a search of the best representation quality among the hidden states within the decoders. For discriminative downstream tasks, task-specific classification heads are implemented on top of the representations of inputs. Positional embeddings can be either truncated to accelerate training or interpolated to enable larger input sequence length. Following PIXEL's work, we demonstrate three settings of fine-tuning as follows to support various needs of downstream tasks:



(1) Word-level rendering

(2) Single sentence rendering

(3) Sentence pair rendering

Figure 3.4. Examples of how we render text for different tasks. For token-level tasks, we render the next token on the next empty image patch. Single sentence rendering is what we use during pre-training. Sentence pair rendering is used for tasks with bi-text.

Token Classification For token-level understanding tasks, each word is rendered to the start of a next new image patch so that each word can be mapped with a unique index of the image patch and word boundary is enabled to avoid that multiple words are overlapped on a single image patch. A linear classifier with dropout is placed on top of these images and the label of each word is assigned only to the first corresponding image patch and cross-entropy loss with softmax is computed on this only during training. Similar fine-tuning formulation is also applied over subword-based and character-based models to assign the label of each word only to is first segmentation.

Sentence Classification For sentence-level understanding tasks, we render the text in the same way as we do in pre-training without adding spaces like in token classification. For bi-text tasks with a hypothesis and a sentence, we concatenate both sentence into one image sequence with a black patch as a separating point. To aggregate the information of the whole sequence, different pooling strategies is applied over the representations of all patches to get the sequence representations, including (1) mean-pooling: compute the mean value of all patches; (2) max-pooling: compute the max value of each dimension of all patches; (3) multi-head attention pooling: average over all patches using the multi-head attention weights. In this paper we mainly focus on the results of method (1), which is proved to be useful in image classification tasks(Liang et al., 2022).

Question Answering For extractive Question Answering tasks like SQuAD(Rajpurkar et al., 2016), the question and the context are rendered in the same way of sentence-pairs as do in sentence classification tasks. Moreover, for overflowing text, we do not truncate the context, instead new examples is created with a stride using sliding window approach. A linear classifier is applied to predict the start and end patches of the answer span.

3.5 Accelerate Training

It took Rust et al. (2022) about 8 days with 8X40GB Nivida A100 GPUs to pre-train PIXEL from scratch for 1 million steps, which is unachievable given the restriction of time and computational resources. Therefore, several techniques that are used to speed up the training time are demonstrated mainly in the following two directions.

3.5.1 Reducing training capacity

First, we used the latest checkpoints of PIXEL-base to initialize our PIXAR's encoder and we only choose to pre-train our model for 0.1 million steps, which is much smaller than 1 million steps. Second, since we only have 16GB Tesla V100 GPUs, we need to first feed the machines with possibly large batch size per device to ensure the converge of gradients used for updating the parameters. The main consumption of machine memory is the weight matrix during the computation of multi-head attentions, but we can still lower the memory cost by reducing the patch size from 16x16 to 8x8 when rendering the text. Also Rust et al. (2022) rendered the text in an RGB style with 3 channels, which we think is unnecessary for representing the text. We choose to render the text with binary pixel values and reduce the channel size from 3 to 1. In combination with these two adjustments, we reduce the rendered pixel values from (3, 16, 8464), indicating (num_channel, patch_size, seq_len×patch_size), to (1, 8, 4232). After preliminary test we found that the maximum batch size per device is now 25 compared to 14 for the original configuration. Therefore, we can use a larger total batch size during pre-training.

3.5.2 Accelerate computing

As proposed by the community to speed up the training of the neural network, mixed precision training is implemented in this project to reduce the computation cost by switch the floating-point values from single precision to half precision for part of the training iterations, which can maximally retain the information with a large speedup in training time. Also, since our model is trained using multiple GPUs on multiple nodes, we implemented distributed data parallel, which is a multiprocessing training scheme that copy the model to each GPU, assign a process to each GPU and gather the gradients together to update the model parameters. This enables us to have a larger batch size with less communication cost within GPUs. Finally, Zero Redundancy Optimizer (Zero) (Rajbhandari et al., 2020) is developed by DeepSpeed to lower the memory and compute demands of each device (GPU) utilized for model training. ZeRO makes use of the aggregate computing and memory resources of data parallelism by distributing the different model training states (weights, gradients, and optimizer states) among the available devices (GPUs and CPUs in the distributed training hardware) to lower the memory usage of each GPU. Zero is developed in three incremental stages, where the latter stages can integrate the ultilisation of the former stages. This project leverage the stage 2 of ZeRo, which partition both the optimizer stage (e.g., for Adam optimizer, 32-bit weights) and 32-bit gradients across the processes. After initial estimation, the batch size per device is almost doubled after the implementation of Zero Stage 2 and also the training time is slightly shorter.

All the training schemes mentioned above are implemented using the hugging-face(Wolf et al., 2019) Trainer, which provides an end-to-end training framework for training neural models and a comprehensive storage of model checkpoints and datasets.

Chapter 4

Experiments

4.1 Preliminary Sanity Check

The first step of our experiment is to adjust the font type used by the text renderer. To investigate how this adjustment will affect the performance of PIXEL and also how we can make the full use of the PIXEL's checkpoints, we construct a series of initial experiments to explore the following two questions:

1. Will changing the font type affect PIXEL's performance in image reconstruction and downstream tasks?
2. How to retrain the model to retain the performance? Can we freeze the transformer blocks and only retrain the embedding and projection layers?

For the convenience of later experiments, we first pre-render the BookCorpus and Wikipedia dataset with the new font type. Then, base on the PIXEL-base checkpoint, we retrain the model in 2 setups: (1) freeze the transformer blocks and only retrain the patch embedding layer in the encoder and linear projection layer in the decoder; (2) retrain the whole model. For simplicity we only retrain the model for 10000 steps. To control the variables for model comparison, we also render the text in grey style and train the model using mean squared error in the same way as (Rust et al., 2022). The reconstruction ability of PIXEL-base and retrained models on the same sample are shown in Figure 4.1.

We find that retrain the embedding and projection layers only is not sufficient in retaining the image reconstruction ability, and that only retraining the whole model can model learn to reconstruct the image patches with the new font type. Then we select two downstream tasks (part-of-speech tagging and Sentiment classification) to evaluate

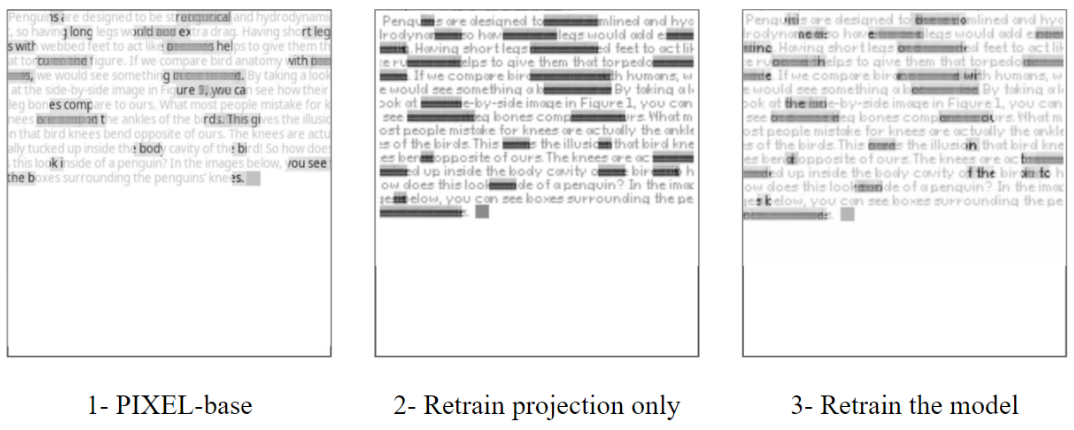


Figure 4.1. Comparison of PIXEL-base and retrained models on the reconstruction ability.

the performance of retrained models. We choose to skip the model with setup (1) for its poor performance in reconstructing image patches. The results of PIXEL-base and retrained model with setup (2) are shown in Table 4.1.

Model	**POS**	**SST2**
	Acc	F1
PIXEL-base	**96.8**	**88.5**
Retrain the model	92.8	82.1

Table 4.1: Comparison of PIXEL-base and retrained model on the performance in part-of-speech tagging and sentiment classification.

The results show that the retrained model still preserve the performance on downstream tasks. Since both the reconstruction ability and performance in downstream tasks are reasonablely good by retraining the whole model for only 10000 steps, we can reach the initial conclusion that changing the font type will not affect PIXEL's ability and we need to retrain the whole model to make the full use of the PIXEL-base checkpoint. This initial experiments also validate the finding of PIXEL that pixel-based language models can transfer to other writing systems.

4.2 Tasks

For comparison, we mainly select the tasks that is used by PIXEL as our downstream tasks. We fine-tune PIXAR on common natural language processing tasks mainly to evaluate its performance in aspects of token-level and sentence-level understanding. Due to the limitations of computational budget and time, we mainly focus on English tasks.

4.2.1 Token-level Understanding

POS We evaluate PIXAR on Part-of-speech (POS) tagging using data from Universal Dependencies V2.10 treebanks (Nivre et al., 2020) for its English subsection. Each word in a sentence is tagged as Noun, Verb , Adjective and so on based on its context and linguistic meaning. The model learns to predict the target tag for each word. The unidirectional way of generating last decoder hidden states make it hard for us to build a root logit and capture bidirectional dependencies for dependency parsing, therefore we neglect it.

NER Also, we evaluate PIXAR's monolingual (ENG) word-level understanding on a named entity recognition (NER) benchmark in ConLL-2003 dataset (Sang and Meulder, 2003). Each word in the dataset is tagged as B-the beginning of a entity, I-the inner of a entity and O-the outside of entities.

We compare how well PIXAR performs on these tasks compared to PIXEL, which allows us to explore the extent to which the auto-regressive way of embedding can retrain the token representation ability.

4.2.2 Sentence-level Understanding

We select several tasks in GLUE, which is a multi-task benchmark for sentence level natural language understanding, for this part of evaluation. We also select a extractive question answering tasks.

SST2 The Stanford Sentiment Treebank(Socher et al., 2013) is a binary classification corpus with fully labeled parse trees that allows for a complete analysis of the effects of sentiment in language.

MNLI Multi-Genre Natural Language Inference(Williams et al., 2018) is a bi-text classification task to determine if a sentence entails, contradicts or is unrelated to a given hypothesis. Both the premise and hypothesis are concatenated into one sequence and be fed into both the encoder and decoder. We present PIXAR to both MNLI-matched, with the validation and test set coming from the same distribution, and MNLI-mismatched, where the validation and test use out-of-domain data.

RTE Recognizing Textual Entailment(Poliak, 2020) is a binary classification task to predict whether a sentence entails a given hypothesis or not.

SQuAD The Stanford Question Answering Dataset(Rajpurkar et al., 2016) is a extractive question answering tasks based on Wikipedia articles. The answer to each question is a segment of text from the corresponding context passage.

These generic tasks provides us with basic knowledge of how Well PIXAR can captures major semantic inference needs across many natural language processing applications, such as Information Retrieval and Question Answering.

4.3 Baseline Model

We compare our results to PIXEL(Rust et al., 2022), a bidirectional masked auto-encoder for language modelling with pixels. PIXEL is a Transformer-based encoder-decoder model trained to reconstruct the masked image patches. Following the work of ViT-MAE, which is designed for computer vision tasks, PIXEL make it suitable for natural language tasks by rendering the text as images for both model input and target. Previous work(Salesky et al., 2021) has demonstrated that the rendered pixels have similar representation power than the text itself. PIXEL is trained by first compressing the unmasked image patches in the dense vector of encoder output and let the decoder to reconstruct the masked image patches based on the encoder output. This is similar to the objective of masked language modelling but in the pixel space. The last encoder hidden states are concatenated with task-specific classification heads to do downstream tasks in the same way as we discussed in section 3.4. Compared with BERT for the same amount of pre-training capacity, PIXEL has shown to perform similarly in both semantic and syntactic tasks but more robust to the text scripts that is not covered in pre-training data, such as multi-lingual texts and noise input.

Due to the restriction of computational cost, we can not try different pre-training configurations for 1 million training steps, which would take us around 8 days according to the estimation. Therefore, our final number of training steps is 0.1M, which is far smaller than PIXEL-base, which makes them less comparable. Our early comparison shows that there is significant performance gap between PIXEL-base and PIXAR. Since PIXEL's author also uploaded their intermediate checkpoints to the huggingface hub for explicit benchmarking, our final baseline model would be PIXEL-base pre-trained for only 0.1M training steps.

Newer monolingual English language models like ROBERTA(Liu et al., 2019) and T5(Raffel et al., 2020) are not included as our baseline models due to their longer pre-training steps and much larger corpus. BART(Lewis et al., 2019) is not included because its intermediate checkpoints are not open-sourced. We follow the same fine-tuning protocols used by Rust et al. (2022) with some adjustments to fit them with the encoder-decoder architecture of PIXAR. The fine-tuning details are listed in AppendixA.2.

4.4 Results

We present the fine-tuning results for both token-level and sentence-level understanding tasks in Table 4.2.

Token-level Understanding We find that PIXAR consistently outperform PIXEL on token classification tasks, especially in NER, where the performance gap is more significant compared to that of POS. This might due to the reason that Named Entity Recognition task has unidirectional nature, the labels of the latter tokens are only dependent on the former tokens, therefore the left-to-right decoding of PIXAR is more efficient here compared to the bidirectional encoding of PIXEL.

Sentence-level Understanding Since our generation unit is image patch instead of token index used by GPT, we can not use either 'CLS' pooling method as PIXEL, nor 'EOS' pooling method as BART. In our early protocol of fine-tuning, we tried different pooling methods, including mean-pooling, max-pooling and multi-head attention pooling. There is not significant performance gap between pooling methods, so we use

	Token-level		Sentence-level			
Model	**POS**	**NER**	**SST2**	**MNLI-M/MM**	**RTE**	**SQuAD**
	Acc	F1	Acc	Acc	Acc	F1
PIXEL-base	93.8	81.2	**82.2**	**69/69.5**	56.3	**69.1**
PIXAR	**94.2**	**82.7**	77.1	68.5/68.9	**57.1**	60.1

Table 4.2: Results for PIXAR and PIXEL fine-tuned for token-level understanding tasks(POS, NER) and sentence-level understanding tasks (SST2, MNLI, RTE, SQuAD). We report test set results over all the tasks. For NER, we report F1-scores due to the serious class imbalance issues among the labels. Over 80 percent of the tokens are labelled with 'O', which makes it meaningless to use accuracy as the metric. For sentence-level classification tasks, mean pooling is used for both models. For the tasks besides SQuAD, both models are fine-tuned for 15000 steps with early stopping. In SQuAD, the fine-tuning step is increased to 20000 due to lower converge speed.

mean pooling here for sentence-level understanding tasks, which is also consistent with the results reported in the PIXEL's paper.

We find that PIXAR reaches contradictory performance on sentence classification tasks. While PIXAR gets comparable or even better results on entailment detection tasks like MNLI and RTE, PIXAR gets much lower scores on SST2 and SQuAD. In our early fine-tuning configuration of SQuAD, we find that PIXEL converges faster than PIXAR at 15000 steps, where PIXEL reaches its minimal training loss and validation loss but PIXAR is still learning towards its converge points. This explains PIXAR's pooer performance on SQuAD. Therefore, we increase the training steps for SQuAD from 15000 to 20000, similar to the configuration of (Rust et al., 2022).

Overall, PIXAR performs similarly or even better in some tasks compared to PIXEL. But PIXEL has significant advantage over PIXAR on SST2 and SQuAD. Since PIXEL was pre-trained on the same corpus with even more seen tokens, we suggest that the encoder-decoder architecture and all the previous adjustments of rendering configurations can retrain or even improve the natural language understanding ability compared to encoder-only architecture of PIXEL at early pre-training steps.

4.5 Ablation study

To further exploit the difference between pre-training configurations, we design a ablation study to evaluate the impacts of model architecture, masking and input representations on the pre-training and downstream tasks.

For model architecture, we compare the encoder-decoder architecture of PIXAR, which use the last decoder hidden states for downstream tasks, to the encoder-only architecture of PIXEL, which use the last encoder hidden states for downstream tasks. However, our encoder-only models are pre-trained in the same way as PIXAR instead of the PIXEL-like pre-training. So in practise, the encoder-only models are just the encoder of their encoder-decoder counterparts. For masking, we compare the span masking and random masking. We also tried to not include any masking at all so that the whole architecture is more like a auto-encoder without any denoising objective. For input representation, we compare the original grey style pixel values of PIXEL, which is a 0 to 1 continuous space, and the binary pixel values of PIXAR, which is a 0 or 1 discrete space. Theoretically, the binary pixel values of a 8×8 grid can represent 2^{64} different inputs, which is definitely larger than the vocabulary size of any existing tokenizers of language models. We tried all the possible combinations of different components and test their performance on the downstream tasks.

The results are shown in Table 4.3. Several findings are clear:

Span masking is crucial for better language understanding Pre-training without masking performs poorly. In our early model protocols, we do not introduce any masking in the encoder to hopefully make use of all the information passed to the encoder. This turns out to be misleading that the model relys too much on the encoder hidden states and just learns to reconstruct the image patches given the encoder hidden states of that specific position instead of predicting new image patches. The best performance is achieved by either random masking or span masking. Generally, span masking outperforms random masking in the downstream tasks.

Last decoder hidden states is a better representation in this pre-training configuration compared with last encoder hidden states Generally, the models with encoder-decoder architecture performs better than the models with encoder-only ar-

				Token-level		Sentence-level			
Model	**Architecture**	**Masking**	**Input type**	**POS**	**NER**	**SST2**	**MNLI-M/MM**	**RTE**	**SQuAD**
				Acc	F1	Acc	Acc	Acc	F1
PIXEL	Enc	Span	Grey	**93.8**	81.2	**82.2**	**69/69.5**	**56.3**	**69.1**
PIXAR	Enc	Random	Binary	93.3	81.4	74.2	65/65.3	53.8	59.7
PIXAR	Enc	Span	Grey	90.1	52.9	57	65.1/65.4	50.9	16.3
PIXAR	Enc	Span	Binary	93.4	**81.5**	75.7	66.3/67.6	54.5	**61.1**
PIXAR	Enc-Dec	-	Binary	92.2	73.1	72.5	57.3/58	49.5	24.1
PIXAR	Enc-Dec	Random	Binary	93.5	79.7	**78.9**	67.9/68.6	53.8	33.6
PIXAR	Enc-Dec	Span	Grey	44.6	45.5	78.4	66.8/67.4	49.1	15.5
PIXAR	Enc-Dec	Span	Binary	**94.2**	**82.7**	77.1	**68.5/68.9**	**57.1**	60.1

Table 4.3: Results of models with different pre-training configurations. The comparisons are made between architectures (encoder-decoder or encoder-only), masking (span masking, random masking or no masking) and input type (grey-style or binary). All models are pre-training using the same corpus for 0.1M steps. We report their test set results over all the downstream tasks.

chitecture. Since our encoder is initialised from the PIXEL-base checkpoints that is pre-trained for 1M steps, the encoder-only models are theoretically pre-training longer than their PIXEL counterpart and are expected to reach better performance. In practise, the adjustment of the pre-training objective and input representation, i.e. changing the font type, makes them fail to outperform PIXEL. However, there are several outliers like model with span masking and grey-style input type, whose encoder-only models performs better than its encoder-decoder counterpart in some of the tasks.

Binary pixel value leads to more stable performance In the framework of PIXAR, the models with grey-style input performs much poorly than the binary models in some of the tasks. We consider it as a stuck in the local minima during training and tried other random seeds and larger learning rates, only to find that it still performs poorly somehow, especially in SQuAD and NER. On the other hand, binary models perform consistently good in all the tasks.

Encoder-only architecture is cruial for SQuAD In the column of SQuAD, we find that encoder-only models always outperform their encoder-decoder counterparts with a large advantage in the F1-scores. This might explain the reason why PIXAR fails to reach similar results as PIXEL in SQuAD.

PIXAR pre-training configuration achieves the most consistently strong performance Among all the models with the framework of PIXAR, the model with encoder-decoder architecture, span masking and binary input type performs well on all tasks with the exception of SST2.

Chapter 5

Conclusions

5.1 limitation

This paper adjust the architecture of PIXEL(Rust et al., 2022) to explore the possibility of generative decoding and also simplifying the input representation. While PIXAR shows promising results for the early pre-training step, there are limitations that are listed for the directions of future works:

- The preliminary objective of this paper is to the realize the generative ability of pixel-based language models, but there is still a long way to go before using pixel-base language models for generative tasks like summarization or abstractive question answering.
  For model architecture, besides the encoder-decoder architecture used by PIXAR, some other frameworks are under exploration. For example, decoder-only models like GPT might be adapted to model pixel-based language modeling by use the conditioned image patches as prefix to predict the following patches. Li et al. (2023) tried to leverage the diffusion framework to model the conditional generation tasks as high-quality image generation condition on the text embeddings.
  Moreover, we are still finding a suitable way to ground the generated images back to the texts. There are out-of-shelf optical character recognizers (OCR) that can be used to do text grounding, but the generation performance might be degraded since OCRs can only detect characters and can not generate fluent texts. Li et al. (2023) develops a solution to this issue by adding a text-grounding block, which is consisted of some transformer layers, above the generate images to mapping them back to the whole sentences. Also we might skip the image generation to directly generated texts in the decoder to avoid the loss that is incurred by both

image generation and text generation.
Eventually, for generation strategy, we usually apply bean search or top K filtering to increase the quality of text generation. However, such strategies are difficult to be implemented for image generation, where their is not a distribution over discrete tokens. Therefore, our implementation of patch-wise image generation is more like a greedy search, which might be bad for generative tasks.

- To simplify the input representation, we use the 8×8 grid and binary pixel values, which is much easier than the 16×16 grid and grey style pixel values used by Rust et al. (2022). However, since we can not redo the pre-training of PIXEL due to the restriction of computational resources, it is still under estimation how the rendering configuration will affect the performance of language models. In the architecture of PIXAR, binary pixel values outperform others, the adjustment of font type is not taken into consideration because the original font type used by PIXEL is not designed over the grid of 8×8, and it is not recognizable when it is fit into a smaller grid. Also, we can use vector quantization(van den Oord et al., 2018) to transform the high resolution images into latent space to further simplify the input representation.

5.2 Conclusion

Pixel-based language model is a new research direction for people to find out whether we could dig out the latent representation power of visual writing system. This could be a alternative to text-based tokenizers with stronger robustness to noisy inputs and more generalization ability to multi-lingual tasks. Theoretically we could embed any scripts that could be rendered as a image. The representation power of pixel is validated by (Rust et al., 2022), but we are still unknown how to do generative tasks with pixel-based language model. Both text and image generation methods should be taken into consideration.

This paper introduced PIXAR, a pixel-based auto-regressive language model. It is a sequence-to-sequence transformer with a bidirectional encoder and a left-to-right decoder. PIXAR is a vocabulary-free model that uses the rendered image of text as input, and produce representational embeddings in the decoder auto-regressively. Also to simplify the input representation, PIXAR choose a smaller patch size and use a different font type that is designed over the 8×8 grids. Initial experiments show that the adjustment of font type won't ruin the performance of PIXEL. This also indicates that pixel-based language model can transfer to new writing systems easily. PIXAR is pre-trained on the rendered version of Bookcorpus and Wikipedia and evaluated on the token-level understanding tasks (POS, NER) and sentence-level understanding tasks(SST2, MNLI, RTE) and extractive question answering tasks (SQuAD). Compared to PIXEL with the same amount of pre-training steps, PIXAR are shown to achieve stronger performance in token-level understanding tasks and similarly or slightly worse in sentene-level understanding tasks. This demonstrates that the encoder-decoder architecture has the same representational power as the encoder-only architecture. The unidirectional way of generating decoder logtis enables PIXAR with stronger ability to represent tokens but slightly poor ability to represent sentence when the comprehensive knowledge of the whole sequence is needed. We also conduct ablation study to validate that the model performs better with span masking and encoder-decoder architecture in our pre-training formulation. And the binary pixel values make the model get stably better scores in the downstream tasks compared to grey-style pixel values. In future work, we will keep working on developing the path for PIXAR to do generative tasks.

Bibliography

A. Baade, P. Peng, and D. Harwath. Mae-ast: Masked autoencoding audio spectrogram transformer, 2022.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL `https://arxiv.org/abs/2005.14165`.

M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR, 2020.

B. Cheng, A. Schwing, and A. Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34: 17864–17875, 2021.

A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale, 2020.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

C. Feichtenhofer, H. Fan, Y. Li, and K. He. Masked autoencoders as spatiotemporal learners, 2022.

P. Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994.

L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021. URL `https://arxiv.org/abs/2101.00027`.

X. Geng, H. Liu, L. Lee, D. Schuurmans, S. Levine, and P. Abbeel. Multimodal masked autoencoders learn transferable representations, 2022.

K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners, 2021.

M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, 8:64–77, 2020.

O. Keren, T. Avinari, R. Tsarfaty, and O. Levy. Breaking character: Are subwords good enough for mrls after all?, 2022.

D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2022.

A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

T. Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.

G. Lample and A. Conneau. Cross-lingual language model pretraining, 2019.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020.

M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

J. Li, W. X. Zhao, J.-Y. Nie, and J.-R. Wen. Glyphdiffusion: Text generation as image generation, 2023.

F. Liang, Y. Li, and D. Marculescu. Supmae: Supervised masked autoencoders are efficient vision learners, 2022.

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2018.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization, 2019.

E. Mansimov, M. Stern, M. Chen, O. Firat, J. Uszkoreit, and P. Jain. Towards end-to-end in-image neural machine translation, 2020.

Y. Meng, W. Wu, F. Wang, X. Li, P. Nie, F. Yin, M. Li, Q. Han, X. Sun, and J. Li. Glyce: Glyph-vectors for chinese character representations, 2020.

J. Nivre, M.-C. de Marneffe, F. Ginter, J. Hajič, C. D. Manning, S. Pyysalo, S. Schuster, F. Tyers, and D. Zeman. Universal dependencies v2: An evergrowing multilingual treebank collection, 2020.

OpenAI. Gpt-4 technical report, 2023.

A. Poliak. A survey on recognizing textual entailment as an nlp evaluation, 2020.

A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. Zero: Memory optimizations toward training trillion parameter models, 2020.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation, 2021.

P. Rust, J. Pfeiffer, I. Vulić, S. Ruder, and I. Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models. *arXiv preprint arXiv:2012.15613*, 2020.

P. Rust, J. F. Lotz, E. Bugliarello, E. Salesky, M. de Lhoneux, and D. Elliott. Language modelling with pixels, 2022.

E. Salesky, D. Etter, and M. Post. Robust open-vocabulary translation from visual text representations, 2021.

E. F. T. K. Sang and F. D. Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition, 2003.

M. Schuster and K. Nakajima. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE, 2012.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL `https://aclanthology.org/P16-1162`.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. URL `https://aclanthology.org/D13-1170`.

K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mass: Masked sequence to sequence pre-training for language generation, 2019.

Z. Sun, X. Li, X. Sun, Y. Meng, X. Ao, Q. He, F. Wu, and J. Li. ChineseBERT: Chinese pretraining enhanced by glyph and Pinyin information. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2065–2075, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.161. URL `https://aclanthology.org/2021.acl-long.161`.

A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning, 2018.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 1096–1103. ACM, 2008. doi: 10.1145/1390156.1390294. URL `https://doi.org/10.1145/1390156.1390294`.

H. Wang, P. Zhang, and E. P. Xing. Word shape matters: Robust machine translation with visual embedding, 2020.

A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference, 2018.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models, 2022.

Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015.

Appendix A

Architecture and Training Details

A.1 Pre-training Details

PARAMETER	VALUE
Image size	(8, 4032, 1)
Patch size	8
Encoder hidden size	768
Encoder intermediate size	3072
Encoder num attention heads	12
Encoder num layers	12
Decoder hidden size	768
Decoder intermediate size	3072
Decoder num attention heads	12
Decoder num layers	12
Layer norm	1e-12
Span masking ratiro	0.25
Span masking max length	6
Dropout probability	0.1
Hidden activation	GeLU
Optimizer	AdamW
Weight decay	0.05
Peak learning rate	1.5e-4
Learning rate schedule	Cosine Decay
Minimum learning rate	0.05
Training steps	0.1M
Batch size	256

Table A.1: PIXAR pre-training settings

A.2 Finetuning Details

PARAMETER	POS	NER	SST2	MNLI	RTE	SQuAD
Rendering backend	PyGame	PyGame	PyGame	PyGame	PyGame	PangoCairo
Pooling Head	-	-	Mean	Mean	Mean	-
Weight decay	0					
Learning rate	3e-5					
Learning rate schedule	Linear decay					
Max sequence length	256	256	256	256	256	400
Batch size	256	256	256	256	256	64
Max steps	15000	15000	15000	15000	15000	20000
Early stopping	True					
Dropout	0.1					

Table A.2: PIXAR finetuning settings