Scaling Pixel-based Language Models

Chen Hu

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2024

Abstract

Recently, Pixel-based Language Models have been shown to perform well in natural language processing tasks. However, most of these pixel-based LLMs can only process discriminative tasks. While PIXAR can handle generative tasks, since the pretraining dataset is English-based, it only attempts English generative tasks. Therefore, this thesis proposed PIXAR++, which can handle seven languages and process and generate images with larger patch sizes. This project attempts more downstream tasks on discriminative tasks and generative tasks in multiple languages. It turns out that PIXAR++ works well in other languages as well. This thesis analyzes the reasons behind the wrong and correct generated text patches and proposes more directions for the development of PIXAR++ models.

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.

(*Chen Hu*)

Acknowledgements

I would like to thank my Supervisors, Prof Antonio Vergari and Prof Suglia Alessandro for all their help, encouragement, and advice. After each meeting with them, I got a deeper understanding of the project and a clearer direction for the next work. I would also like to thank Yintao Tai, my senior, for helping me with this project. Since my project is scaling his model, he gave me advice and help in many stages of my project.

Table of Contents

Chapter 1

Introduction and Motivation

1.1 Introduction

In traditional natural language processing (NLP) research, the selection and use of tokenizers affect many aspects of model training. Tokenizers are designed to segment text into a sequence of small sub-units. Common sub-units include sentence pieces, bytes, characters, sub-words, or words. However, the vocabularies of these sub-units often require a lot of effort to create and maintain [1]. Besides, these vocabularies also have particular limitations.

For a vocabulary consisting of word units, it is not possible to include all the words in any one language in advance. This is because collecting all the uncommon words before the training of NLP models is difficult, and new words are created in the daily conversation and writing of people. Therefore, out-of-vocabulary (OOV) words always exist. According to the research of Kaddour et al., the performance of NLP models was heavily decreased when encountering these OOVs [2]. Besides, because bytes and characters are very small, the sequence made up of them can be very long. A long sequence increases the burden on the embedding and output layers of the model [3]. The sub-word is more flexible. It can build vocabularies of different sizes as required, which relieves the burden on both the embedding and output layers, but also creates a dilemma. Although this kind of vocabulary performs well in the training of a single language, when faced with multilingual datasets, the researchers need to either expand the size of the vocabulary or fix the number of sub-words that the vocabulary can contain. Therefore, a fixed vocabulary is bound to have limitations. This limitation mainly exists in the encoding of input and the calculation of the probability distribution of vocabularies [4].

According to the research of Rayner et al., when facing incorrect words or unusual sentence structures, humans can still make sense of sentences by integrating visual and textual information [5]. This means that the graphic information contained in the text can help people understand the sentences. Therefore, NLP models can also be trained by learning graphic information or pixel information contained in the text [6]. Based on the above research, a Pixel-based Encoder of Language, PIXAL [4] was designed. This model used a sequence of fixed-size rendered patches to replace the embedding layer in the traditional NLP pipelines. This model does not need to process raw text but only needs to extract features from pixels in the patch to learn [4]. However, PIXEL is not proper in dealing with generative tasks. Therefore, the first pixel-based autoregressive large language model (LLM), PIXAR [7], was designed to process the generative tasks of NLP. In the pretraining stage, this model predicts the next patches consisting of pixels only depending on the previously rendered text patches [7]. However, PIXAR can still be extended.

1.2 Motivation

This dissertation aims to scale a pixel-based language model, PIXAR. It is the first pixel-based autoregressive LLM [7]. According to the introduction, the traditional NLP model only focuses on the text information but ignores the graphic information of the text. For humans, however, the graphic information of a text is a necessary component in helping people understand sentences. Therefore, it is worth learning and generating text using the pixel information of text images as model input. Since PIXAR has already done some encouraging research in this field, this project decided to scale PIXAR in the following ways.

Firstly, the patch of the input and output images are fixed in size. However, increasing patch size can increase the length of the text sequence contained in each patch, so that the model can generate a longer text sequence on the premise that the number of generated patches is fixed. Besides, PIXAR is only tried on the English dataset and all the experiments applied to PIXAR are based on English. To expand the scope and robustness of PIXAR, this thesis will attempt to expand PIXAR into a multilingual version. Finally, according to the experiment results of Dhariwal et al., diffusion models always have better performance than the GAN models [8]. Therefore, in further research, the Diffusion transformer or Diffusion models will be used to replace the GAN as the final layer of the PIXAR model [8] [9] [10].

The development of this PIXAR++ has many benefits both for NLP research and applications. First, for NLP research, pixel information was proved to be useful for model training. Therefore, these useful graphical features can be utilized and even combined with traditional NLP training, so that the trained model can more accurately understand the meaning of a sentence. In addition, if the model only uses text images as its input, it will be free from the constraints of the fixed vocabulary. When this model is faced with OOV, the performance of it will not drop too much. Besides, because this study connects NLP and CV, it has a broad development prospect. Finally, the expansion of language types and patch size enables PIXAR++ to have more application scope and stronger generalization ability.

1.3 Objective

This project aims to extend PIXAR. Since PIXAR has done experiments on English, the new model PXIAR++ will focus on multilingual datasets. Besides, a larger patch size is also tried on PIXAR++. **The project has the following main objectives:**

- Environment configuration and preprocessing the dataset.
- Training PIXAR stage one: MLE.
- Training PIXAR stage two: Adversarial.
- Designing a multilingual dataset for PIXAR++.
- Training PIXAR++ on this multilingual dataset.
- Training PIXAR++ on a larger patch size.
- Evaluating this trained PIXAR++ model.
- Finish dissertation.

1.4 Results achieved

Based on background technology and motivation, this thesis trained two models named 8-patch-size PIXAR++ and 16-patch-size PIXAR++ (The input and output patch size is 8 * 8 or 16 * 16) that can deal with multilingual language discriminative and generative tasks. Besides, the training process includes two stages, the trained models in each

stage are tried on all the downstream tasks. According to the paper on PIXAR, since the patch size used in PIXAR is 8 * 8 [7], the comparison between PIXAR and 8-patch-size PIXAR++ can better reflect the advantages and disadvantages of PIXAR++.

Discriminative tasks: The performance of 8-patch-size PIXAR++ on the GLUE benchmark is slightly lower than PIXAR. Since this task is pure English, this result is in the assumption. However, specifically, 8-patch-size PIXAR++$_{stage1}$ even does better on RTE and WNLI, and 8-patch-size PIXAR++$_{stage2}$ performs better on RTR than PIXAR in GLUE benchmark [11]. Besides, the 8-patch-size PIXAR++ performs better than PIXAR on most of the XNLI tasks except English and Turkish [12]. Besides, 16-patch-size PIXAR++ performs worse on every discriminative task of the other two models.

Generative tasks: Since the training dataset of PIXAR is in pure English, the downstream tasks it attempts contain only bAbI and the English LAMBADA [7]. PIXAR is superior to PIXAR++ in the accuracy of generating text. Only on the readability of the Stage 1 models, 8-patch-size PIXAR++ is superior to PIXAR. This means that PIXAR is still better at English generation tasks than PIXAR++ and that GAN loss improves the readability of single-language models more than multilingual models [7] [10]. Besides, for LAMBADA tasks in other languages, both PIXAR++ models also have similar performance to English LAMBADA.

1.5 Thesis structure [1]

- **Introduction and Motivation:** The introduction, motivation, objective, results, and structure of the thesis.
- **Background and Literature Review:** The background research related to this thesis and the technical basis related to PIXAR++.
- **Methodology:** The core preprocessing, training, and evaluation methods and models used in this thesis.
- **Experiments and Results:** The preprocess, training, and the performance of the experiments baselines and PIXAR++ on downstream tasks.
- **Conclusion and Discussion:** The achievements, limitations, and future works are mentioned in this chapter.

[1] Part of the first three chapters of this thesis are paraphrased from ipp.

Chapter 2

Background and Literature Review

2.1 Language Models

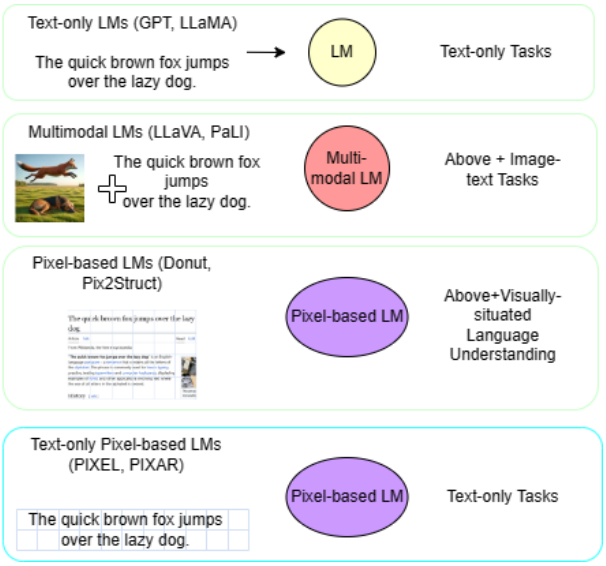


Figure 2.1: This image shows the classification of LM according to the kind of input data [13]

As shown in Figure 1, currently LMs can be divided into four categories according to the type of input data. The first is Text-only LMs that only take texts as input, including GPT [14] and LLaMA [15]. The second is Multimodal LMs that take images and texts as input, represented by LLaVA [16] and PALI [17]. The third is the pixel-based LMs that take not text-only images as input, including Donut [18] and Pix2Struct [19]. The last one is the pixel-based LMs that take text-only images as input, which is mainly studied in this thesis. The main representative models are PIXEL [4], PIXER [7], and

PTP [13]. Therefore, the following sections will review recent pixel-based LMs that take images as input.

Since Chinese is a pictograph, its characters contain a lot of graphic information in the text, and most of the early Pixel-based natural language processing models used Chinese datasets as their input. One of the earliest experiments was done by Liu et al., a CNN-based model was used in their project to extract the graphic information of handwritten Chinese characters on character-level and then the similarity of these Chinese characters was compared [20]. Besides, according to the research of Sun et al., the classification tasks of Chinese characters were also tried through training models on character-based datasets [21]. Traditional symbolic tokenizers miss out on a lot of graphic information in Chinese characters, but the Tianzige features that highlight graphic information in Chinese characters can be well utilized [22] [23]. Besides, character-level graphic features are integrated into the ChineseBERT embedding vector to train Bert-based models [6]. However, character-based models still have some limitations. First, since the smallest unit of a picture is not a pixel but a character, these models did not capture all the information in the picture. In addition, some emojis will also affect the performance of such models. Based on the above research, pixel-based models will capture more comprehensive visual information about text, which was designed to address these limitations.

When talking about Pixel-based models, the first model to introduce is designed by Salesky et al. [24] which uses visual text datasets as its inputs. This model is used to solve machine translation tasks. However, the embeddings of a fixed vocabulary depended on the output layer of this model, which means it is not a pure pixel-based model [24]. According to the idea of Salesky et al., PIXEL (Pixel-based Encoder of Language), the first complete LLM (large language model) using pixels of images as input was designed based on the Masked Autoencoding Visual Transformer (ViT-MAE) [25]. On the basis of the transformer model, after training the encoder-decoder model ViT-MAE, the pixels of the masked image patches were reformed. This model gets rid of the restricted vocabulary embedding layer. As the replacement, The raw text was rendered into a sequence of fixed-sized patches by using a Vision Transformer encoder [26]. PIXEL contains three main parts, renderer, encoder, and decoder. The renderer turns texts into images, the unmasked parts of the image are encoded through an encoder, and the decoder reforms the masked parts [4]. To expand the application of PIXEL, multiple language datasets were tried as the input of PIXEL in 2023 [27]. Besides, instead of using the text encoder of CLIP based on ID, CLIPPO solves visual

QA tasks through the process of both images of rendered texts and normal images [28] [29]. This research makes the connection between NLP and CV, which means the multimodal models can be trained by both images and texts. Since both PIXEL and CLIPPO contain the encoder part in their architecture, they cannot be trained to solve generative tasks. Using encoded text features as conditions, a diffusion model was used by GlyphDiffusion to generate images of texts from noise [30]. However, the embedding of symbols was still applied in this model. Therefore, according to the design of LLMs only based on pixels, a model named PIXAR was created to learn the representation of symbols by only processing perceptual information [7].

PIXAR (PIXel-based AutoRegressive LLM) was designed based on the PIXEL model to process the generative tasks that PIXEL can not handle[7]. MAE structure used in PIXEL was replaced by other generative LLMs including LLaMA-2 and GPT-2 [15]. The input and output of this model are only patches of pixels with text information on them. In the pretraining stage, this model generates new image patches only learned from the previous image patches. In the second finetune stage, this model chooses the GAN loss as the final layer and this model combines the MSE loss and GAN loss for RGB images and combines pixel-wise binary cross entropy loss and GAN loss for binary images. Another model named PTP (Patch-and-Text Prediction) [13] was also designed conditioned on PIXAR. This model contains both image and text decoders, which means this model can be used to predict not only the image with masked text content but also the pure text content from the text images. As shown in the paper on PTP, for the GLUE benchmark, PTP performs better than PIXAR and PIXEL on every task of GLUE, which means PTP has better performance on discriminative tasks. However, since PTP did not try to do the experiments on the LAMBADA and bAbI tasks, the generative ability of PIXAR and PTP has not been compared [13].

PIXAR Preprocessing: Since this project will use PIXAR as the base model, here will introduce the implementation of PIXAR in detail. Since PIXEL is designed based on PIXAR, the preprocessing part of them is similar. Firstly, the articles of the raw datasets were divided into small paragraphs within a fixed number of characters using the ”PunktSentenceTokenizer” from the Natural Language Toolkit (NLTK). As shown in Figure 2.2, These small paragraphs are treated as the input text. After that, a long (single) image consisting of several nonoverlapping patches is generated to represent these small paragraphs. In the experiments of PIXAR, the binary images ($x \in [0,1]^{H\times W\times 1}$) and RGB images ($x \in [0,1]^{H\times W\times 3}$) were tried. These images are made up of patches with a fixed size. A vector is generated by flatting each patch of

the input image $x \in R^{H \times W \times C}$ (H: the patch height; W: the patch width; C: number of channels). A hidden embedding $h^{in} \in R^d$ is then created by projecting the vector. All the patch embeddings have resulted in a sequence of $\{h_1^{in}, ..., h_{eos}^{in}\}$ which is used to be the input of the transformer decoder block. According to the experiment of the PIXAR paper, binary images not only simplified model learning, but also obtained relatively good downstream task performance [7].

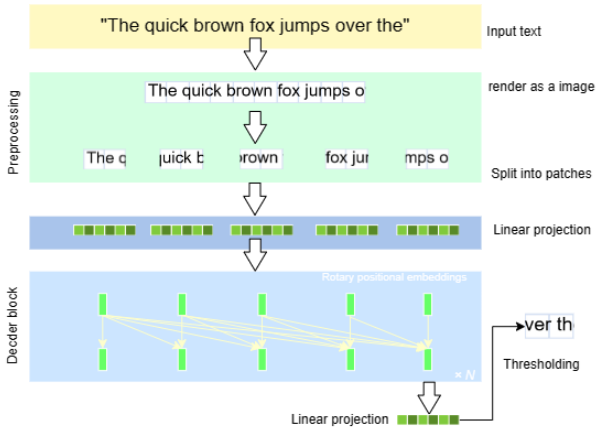


Figure 2.2: This image shows the preprocessing and decoder process of PIXAR

PIXAR structure: Unlike the PIXEL model, PIXAR is a decoder-only model. A stack of 12 Transformer layers is contained in its decoder. Aiming to increase the performance of the transformer used in this model, pre-normalization using RMSNorm [31], SwiGLU activation functions [32], LLaMA-2 [15] and rotary positional embeddings [33] are proposed to be used in PIXAR. The output generated by these transformer layers in PIXAR is $h^{out} \in R^d$ hidden states [7]. Finally, new image patches are generated as the output of the model. The specific process is to add a linear layer after the transformer layer, which can map the output embedding h_N^{out} back to the space of the pixel as a vector. This vector named $\tilde{x}$ can be represented as the linearized $H \times W \times C$ (For binary image C is equal to 1) patches and can be interpreted according to the category of the image. Setting the temperature to $T = 1$, an element-wise sigmoid squashed the vector $\tilde{x}$ for the rendered binary images, where $\tilde{x}$ are the logits. Besides, a threshold $\theta = 0.5$ was applied to generate a hard binary vector for the original vector with the values of the probabilities which is between $[0, 1]$. For the processing progress of the RGB images, the value of the $\tilde{x}$ element-wise is firstly clipped to be within $[0, 1]$. After that, the RGB patches are created by linearly mapping the three channels in the $\tilde{x}$ to $\{0 - 255\}$ [7].

Training stages and loss functions: There are two stages in the training of PIXAR

which are **Stage 1 training: MLE** (maximum likelihood estimation) and **Stage 2 training: Adversarial**. In stage 1 training, a gold (observed) patches sequence ($x_{1:i-1}$) and a sequence of L ground truth pixel patches ($x_{i:i+L-1}$) are prepared to calculate the negative log-likelihood. The negative log-likelihood of $x_{i:i+L-1}$ conditioned on $x_{1:i-1}$ is minimized to calculate the MLE, which is named "teacher forcing" [34]. Given the embedding of the last layer h_N^{out}, the pixels in $x_{i:i+L-1}$ are considered to be conditionally independent. According to the assumption above, the reconstruction loss L_{rec} over $x_{i:i+L-1}$ is minimized. For the RGB images, the MSE loss is applied, and for the binary images, the usual pixel-wise binary cross-entropy loss is used [7] [35] [36]. Since there are $H \times W \times C \times L$ variables in the sequential prediction task, this task is challenging for the PIXAR model to predict. Although the choice of using binary images relieves the predicted budget of the learning, the pretrained PIXAR very easily generates patches with noise which is more likely to occur when L>1, and always gets stuck in the local optimal. According to the research of Theis et al., MLE tends to insert probabilistic mass into possible modes which means the circumstance above can be predicted [37]. Therefore, the first stage of training will lead to low readability of the generated patches [7].

To solve the above problems, PIXAR paper modified the original L_{rec}. An adversarial loss is added to the original loss function. The newly added loss function is named patch-wise context-aware adversarial (PCAA) loss. Besides, the generation performance and readability of PIXAR can be greatly increased by only 200 steps of stage two training. Based on the basic structure of the GAN [10] model, both a discriminator and a generator should be used for the adversarial training. The PCAA Loss Function is:

$$L_{PCAA} = E_{\tilde{x}_{i:i+L-1}}[-log(D(\tilde{x}_{i:i+L-1}|x_{1:i-1}))] \tag{2.1}$$

Where $\tilde{x}_{i:i+L-1}$ is the ground truth pixel patches; $x_{1:i-1}$ is the observed patches.

The usage of this equation is to measure how much the discriminator can be "fooled" by the generator by letting the discriminator "guess" whether a generated patch $\tilde{x}_i$ is real or fake. A copy of the stage one PIXAR is used with a patch-wise classification head to be a context-aware discriminator. This discriminator is used to compute the PCAA loss. Based on the real patches $x_{1:i-1}$ provided, the training of this discriminator is to justify whether the input patch is fake or real [7].

A patch sampling algorithm is designed to compute the PCCA loss effectively and reduce the computational burden of the transformer layers. The fake patches are first

generated given a sequence of patches and the reconstruction loss is calculated by the generator. After that, the key and value vectors of the real patches and the PCAA loss of each fake patch are calculated by the discriminator. Only 30 sampled fake patches are used to compute the PCAA loss for a sequence to increase the training speed of stage 2 [7].

The last step is to choose the hyperparameters to balance the PCAA and MLE. Because GAN training is extremely unstable, according to the [38] study, the paper of PIXAR decided to mix MLE loss (L_{rec}) and PCAA loss (L_{PCAA}). The following is the loss function of stage 2:

$$L_{com} = L_{rec} + \lambda_m \cdot \lambda_{auto} \cdot L_{PCAA} \tag{2.2}$$

where λ_m is a hyperparameter that can be manually modified, and $\lambda_{auto} = \nabla_{G_L}[L_rec] / (\nabla_{G_L}[L_{PCAA} + \delta])$ where $\nabla_{G_L}[\cdot]$ is the scale of gradients of the generator related to the last layer, and the $\delta = 1e^{-8}$ is used to protect this equation by avoiding division by zero.

2.2 Diffusion Models

The basic theory of the diffusion model is based on non-equilibrium thermodynamics. The diffusion process begins with random noise and data resembling real videos and images are generated by removing this noise. The idea of using the diffusion model to generate images starts with Stable Diffusion. In this model, multiple diffusion steps are used to increase the throughput of the model, using a gradual increase in noise to improve the ability of control, so that the model can generate high-quality images [39]. The OpenAI Sora was proposed based on Diffusion Models and Transformers. High-quality video of around 1 minute can be generated by this model [40] [9]. Two models are used in SORA which are Latent Diffusion Models [39] and Diffusion Transformers (DiT; [41]). Latent Diffusion Models have great advantages in the task of compositing high-quality images. The computation cost can be decreased by using the diffusion model in the latent space. The most important part of the SORA is the Diffusion Transformer. The U-Net module of the classic diffusion model was replaced by the transformer to make up the latent diffusion model. Using this structure can speed up the processing efficiency of image patches and reduce the computing resources required to generate high-quality images [41]. In further study, the diffusion model or diffusion transformer is planned to be used as the final layer in PIXAR++ instead of using the GAN loss [7].

Chapter 3

Methodology

3.1 Datasets and downstream tasks

At the beginning of the tasks, environment configuration, data preprocessing, and training the original PIXAR are the main work directions. Two English datasets used in the paper on PIXAR are chosen to retrain PIXAR which are Bookcorpus [42] and English Wikipedia [4] [7]. Because of the lack of training resources and because this project had a PIXAR_{stage2} checkpoint of 85M parameters provided by the PIXAR authors, the original PIXAR model did not need to finish training. Besides, after research, Wikipedia datasets contain other language versions. Therefore, seven Wikipedia datasets with seven different languages including German, French, Spanish, English, Russian, Arabic, and Italian were chosen and combined to create a new multilingual dataset. This dataset was used in the first and second stages of training.

For the downstream tasks, this project chose GLUE benchmark and XNLI as the discriminate tasks and Lambada and bAbI tasks for the experiment on generative tasks [4] [11] [12] [43] [44].

3.1.1 GLUE

GLUE benchmark consists of nine tasks, where eight are classification tasks and only STS-B is regression tasks. According to the type of tasks, it mainly includes three kinds of tasks: similarity and paraphrase tasks, inference tasks, and single-sentence tasks [11]. Most of the tasks use accuracy as their metric and other metrics will be mentioned specifically in the task introduction below.

Single-sentence tasks:

- **CoLA (Corpus of Linguistic Acceptability) [45]:** This task contains judgments of English acceptability derived from articles and books on linguistic theory. Each of these samples is a sequence of words that the language model needs to determine if it is grammatically correct. The evaluation metric used here is Matthews correlation coefficient [46]. It presents the performance of the unbalanced binary classification. The value of it is from -1 to 1, and the value 0 is guessing without being informed [45].

- **SST-2 (Stanford Sentiment Treebank) [47]:** This evaluation task contains the sentiment annotated by humans and the movie reviews. The language models need to predict whether the sentiment of sentences provided is positive or negative [47].

Similarity and paraphrase tasks:

- **MRPC (Microsoft Research Paraphrase Corpus) [48]:** This task contains many sentence pairs taken from online news sources automatically. These sentences have been manually marked for semantically equivalent or not. Since it is an unbalanced dataset, this project reported an F1 score for this task [48].

- **QQP (Quora Question Pairs):** The QQP dataset contains a set of question pairs extracted from the corpus of community Q&A websites. Like MRPC, it is also a downstream task to test whether two sentences are semantically equivalent. It is also a task with an unbalanced dataset. Therefore, the F1 score is also applied in this project to evaluate the performance of the model on this task [11].

- **STS-B (Semantic Textual Similarity Benchmark) [49]:** The STS-B consists of some sentence pairs extracting image and video titles, news headlines, and the inference data of natural language. Each sentence pair was manually labeled with a score from 1 to 5 based on similarity. This task aims to predict these scores and the performance of the language model is evaluated by Pearson and Spearman correlation coefficients [49].

Inference tasks:

- **MNLI (Multi-Genre Natural Language Inference Corpus) [50]:** The collection of sentence pairs in this task is crowd-sourced with the annotations of textual entailment. Each sentence pair consists of a premise sentence and a hypothesis

sentence. The language model needs to predict whether the hypothesis is contradicted by the premise (contradiction), the hypothesis is entailed by the premise (entailment), or neither (neutral). Ten different sources were used to collect the premise sentences. The sources include fiction, government reports, and speech. The language models need to predict the matched (in-domain) and mismatched (cross-domain) sets [50].

- **QNLI (Stanford Question Answering Dataset) [51]:** It is a Q&A tasks with pairs of a question and a paragraph. The questions in this task were designed by the annotator and the answers are one of the sentences in the Wikipedia paragraph. The articles in this dataset are broken down into many sentences, each sentence is combined with the corresponding question in the dataset to form sentence pairs, and those with poor lexical matching are removed. This modified version changes the searching answer task into a classification task and removes the assumption that the answer always exists in the second sentence. Besides, the new task considers the lexical overlap an important clue [51].

- **RTE (Recognizing Textual Entailment):** This task is a collection of four annual text entailment problem challenges including RTE1, RTE2, RTE3, and RTE5. The samples included are collected from Wikipedia text and news. All the datasets are converted into binary classification problems. For the three classification datasets, the contradiction and neutral are changed into not entailment [11].

- **WNLI (Winograd Schema Challenge) [52]:** This is a reading comprehension task where a sentence with a pronoun is read and the referent to which the pronoun refers is selected. To turn this task into a classification task, pronouns are replaced with alternative referents. The model needs to determine whether there is an entailment relationship between two sentences. The main source of the dataset is fiction books [52].

3.1.2 XNLI (Cross-lingual Natural Language Inference corpus) task:

This task extends the test and development examples of the Multi-Genre Natural Language Inference Corpus (MultiNLI) to 15 languages to build an evaluation set for cross-lingual language understanding (XLU). it contains 7500 development and test samples annotated by humans in the three class classification of natural language inference (NLI) in Bulgarian, English, French, Spanish, Greek, German, Vietnamese,

Turkish, Russian, Arabic, Hindi, Thai, Chinese, Swahili, and Urdu. These languages include several language categories and two low-resource languages, Swahili and Urdu are contained [12].

3.1.3 BAbI task:

There are 20 tasks in the bAbI. All the tasks do not have noise and a hundred percent accuracy can always be achieved by a human that can read English. These tasks are simple and routine for humans and do not require any knowledge background, for example, logic, or machine learning, to solve them. This project chose task one to test the performance of PIXAR++. Task one consists of questions that possibly contain a set of unrelated facts and one supporting fact. The model should find the true result for the question [43].

3.1.4 LAMBADA tasks:

The LAMBADA (LAnguage Modeling Broadened to Account for Discourse Aspects) task is a dataset that assesses the performance of the text comprehension ability of a model. The samples of LAMBADA are narrative passages. One thing these articles have in common is that people can easily guess the last word if provided with the whole passage. If humans only read the last sentence, they can not guess the true answer. Therefore, a model that wants to perform well on LAMBADA needs to learn long-distance dependencies rather than being limited to the local context [53]. Since the training dataset used in this project is a multilingual dataset, generative LAMBADA tasks in other languages including German, French, Spanish, and Italian will also be used [44].

3.2 Preprocessing

In the preprocessing stage, the main task is to preprocess the multi-language dataset created in the previous section. Because Wikipedia articles vary in length, they sometimes go beyond the input window of PIXAR. Therefore, this project uses the same algorithm as PIXAR to segment these articles. The method named “PunktSentenceTokenizer” was chosen from the Natural Language Toolkit (NLTK) to divide these articles into sentences [54]. These sentences are combined into small paragraphs within a fixed number of characters. After that, the samples that contain characters less than 100 are

deleted. The window size of the PIXAR++ is 360 patches and the character limit is 1180 characters which is the same as PIXAR [7]. According to Table 3.1, the English Wikipedia has the most average characters which is 6295 and the Arabic Wikipedia has the lowest average characters which is 1283. Although the differences between the two languages are significant, this multilingual dataset contains, in the greatest likelihood, the most text in English. In addition, the average characters of the preprocessed dataset were 960 and the total samples of it were 27,138,373, which was similar to those of PIXAR [7].

Dataset	Number of samples	Average characters
Arabic Wikipedia	1024000	1283
English Wikipedia	1024000	6295
French Wikipedia	1024000	3930
German Wikipedia	1024000	4238
Italian Wikipedia	1024000	2900
Russian Wikipedia	1024000	3166
Spanish Wikipedia	1024000	3546
Rendered dataset	27138373	960

Table 3.1: datasets information

According to the paper of PIXAR, the previously processed texts are rendered into a long image containing several patches through PangoCairo render, the same tool used in PIXEL. Based on the experiment results of PIXAR, binary images perform better and relieve the burden of computation. Therefore, PIXAR++ chooses the binary image ($x \in [0,1]^{H \times W \times 1}$) as the input image. Each image is then cropped into small patches with fixed-size ($x \in [0,1]^{8 \times 8 \times 1}$) for 8-patch-size PIXAR++ and ($x \in [0,1]^{16 \times 16 \times 1}$) for 16-patch-size PIXAR++. Due to the good performance of PIXAR, the pixel-style font "Pixeloid Sans" is chosen to generate the input images for PIXAR++. Finally, through the linear projection, these patches are changed into vectors to create a hidden embedding and input into the transformer decoder block [7].

3.3 Training stages

For the pretraining stage 1, since all the images used in PIXAR++ are binary, the usual pixel-wise binary cross-entropy loss is used to pretrain the PIXAR++ model. Same

as the PXIAR model, the reconstruction loss L_{rec} is calculated in this stage. Table 3.2 shows the structure of the PIXAR++ model and the hyperparameters used for stage 1 training. PIXAR++ is also a decoder-only model that contains 12 transformer layers and 12 attention heads. The number of parameters of these two pretrained PIXAR++ models is around 85M which is the same as the PIXAR model. Since the input patch size of the 16-patch-size PIXAR++ is 16 * 16, the parameters of it are a little more than the other model. Most of the hyperparameters are the same as the PIXAR model, the batch size was changed to 768 and the steps were changed to 0.5M to increase the training speed. In addition, during the training period, the learning rate is warmed up to 3e-4 linearly and then annealed to 3e-6 through the cosine scheduler. Table 3.3 shows the new hyperparameters of stage 2 training, the learning rate and GAN learning rate are changed to 3e-6, and the evaluation steps are changed to 100 to store more checkpoints. According to PIXAR, the GAN ratio chosen by the author is from 0.1 to 15. After trying several GAN ratios from this domain, 0.8 is picked for 16-patch-size PIXAR++ and 1.6 is picked for 8-patch-size PIXAR++. The loss function L_{com} mentioned in the methodology is also used in this project [7].

Render Configuration		Model Structure		Pretrain Hyperparameters	
patch length	2	layers	12	peak lr	3e-4
patch number	360	attention heads	12	min. lr	3e-5
min char.	100	hidden size	768	lr scheduler	CosineAnnealing
max char.	1180	activation	SwiGLU	optimizer	AdamW
render DPI	80	intermediate size	2048	β_1	0.9
font size	8	parameters	85.2M / 85.7M	β_2	0.95
patch size	8 / 16			weight decay	0.1
font	PixeloidSans			steps	0.5M
binary	true			warm up	2000
Temperature (T)	1			batch size	768
Threshold (θ)	0.5			precision	fp16 & fp32
				random seed	42

Table 3.2: This table shows the configuration of rendering the original text datasets, the structure of models, and the hyperparameters of the pertaining stage.

3.4 Text recognization

For the generative tasks, it is necessary to recognize the text from the images and check the readability of the generated text. For text recognition, the OCR software is chosen to

Stage 2 Hyperparameters			
lr	3e-6	GAN lr	3e-6
lr scheduler	CosineAnnealing	GAN lr warmup steps	100
optimizer	AdamW	GAN total steps	1000
β_1	0.9	GAN ratio	0.8 / 1.6
β_2	0.95	GAN ratio warmup steps	100
weight decay	0.1	random seed	42
steps	1000	batch size	32
warmup	100		
precision	fp16		
evaluation freq.	100 / 50		

Table 3.3: This table shows the hyperparameters used in the stage of training GAN.

recognize the text from the images. These images are created by putting the generated patches together. The extracted text is more accurate because OCR software performs better on images with higher resolution. In addition, the performance of OCR software on binary images is also poor. Even when faced with words humans can understand, its recognition is still wrong. To improve the accuracy of the recognition, the generated patches were scaled by 3 in size and placed in the middle of the square white background. The Tesseract OCR [1] and Paddle OCR [2] are chosen to recognize the texts from the output images. Because this project trains a multilingual model. Therefore, during the evaluation, the LAMBADA dataset was tested in five different languages. Therefore, this project will explore two different readabilities, namely whether the generated text belongs to the same language as the prompt and whether the generated text belongs to one of the five languages [7].

However, for Paddle OCR, it can only used by specifying one language as its recognizing language. If five Paddle OCRs for different languages are used separately to recognize the output images, the evaluation time will be increased. Besides, because different languages have different alphabet tables, OCR for a single language might recognize a wrong word correctly. For example, an incorrect French word may be recognized by the English paddle OCR as the correct word, since the French alphabet contains some characters like: ”à” which may be recognized to be ”a” by the English Paddle OCR. Therefore, when facing different languages, the Paddle OCR for the same language as the prompt language will be used as the recognizer.

For Tesseract OCR the multilingual version of it will be used as the recognizer. The

[1]Tesseract OCR: link
[2]Paddle OCR: link

reason is that if single-language OCR is used, some wrong words that are more likely to be in other languages may be forced to be recognized as correct words by such OCR due to differences in the alphabet. While the use of multilingual OCR may slightly reduce the accuracy of text recognition, it makes text recognition more rigorous. The results of the Tesseract OCR and Paddle OCR are combined. If one of them recognizes the target word, the prediction is considered correct [7].

Table 3.4 shows the letters that are not included in English but are contained in the other four languages. These letters are copied from Chinese version Wikipedia [3]

language	Letters outside the English alphabet
German	(Ä ä) (Ö ö) (SS ß) (Ü ü)
French	(À à) (Â â) (Æ æ) (Ç ç) (É é) (È è) (Ê ê) (Ë ë) (Î î) (Ï ï)(Ô ô) (Œ œ) (Û û) (Ù ù) (Ü ü) (Ÿ ÿ)
Spanish	(Á á) (Ch ch) (É é) (Í í) (Ll ll) (Ó ó) (Ú ú) (Ü ü)
Italian	(À à) (È è) (É é) (Ì ì) (Í í) (Î î) (Ò ò) (Ó ó)(Ù ù) (Ú ú)

Table 3.4: This table shows the letters outside the English alphabet in the other 4 languages.

.

3.5 Readability metric

The patches of images generated by PIXAR++ may contain readable text or unreadable text, and noise. Whether the output patches are readable for humans and OCR tools or not depends on the generation quality of the PIXAR++ model. The standard for measuring whether the generated text is readable is called readability. Because this project uses two OCR tools as text recognition tools, the readability in this project is defined as whether the generated patches can be recognized as at least one word by the OCR tools. This project used two readability metrics, one is whether the generated word exists in the same language word list as the prompt, and the other is whether the generated word exists in the five languages word list. The languages of the five languages word list are English, French, German, Italian, and Spanish, which are the same languages used in LAMBADA prompts. The reference vocabulary of English is collected by using the English Word Frequency dataset [4], which contains 333k most common English words. For the other four languages, the datasets are chosen from the

[3]Wikipedia (Chinese version): link

[4]English Word Frequency dataset: link

WorldLex [5]. There are two sets for each language which are raw freq. and cleaned freq.. The cleaned freq. datasets were tried first but the performance was not good. The reason is that, for example, the German dataset only contains around 150k common words which is much less than the chosen English dataset. Therefore, the raw freq. datasets in this website were chosen as the vocabulary lists in this project. Since all these datasets are much more than 333k words, this project only used around 333k words on the top of each dataset CSV file [7].

[5]WorldLex: Blog, Twitter and Newspapers Word Frequencies for 66 languages: link

Chapter 4

Experiments and Results

4.1 Data Preprocessing

Because both PIXAR and PIXEL chose Wikipedia datasets for their experiments, this project selected seven different language datasets from Wikipedia including French, English, Spanish, Arabic, Russian, German, and Italian. Chinese was chosen at the beginning of the experiment but performed poorly in the evaluation due to the small font size selected for the experiment. Figure 4.1 shows a rendered image comparing English and Chinese. It can be seen that every character in English can be clearly recognized, while many characters in Chinese are gathered together by many black pixels, which makes it difficult to recognize the text. In addition, there are many Chinese characters, but the dataset is only 1,024,000 articles, so many characters may only appear once. Therefore, Chinese was replaced with Spanish.

Besides, the parameters of PIXAR++ specified in pretraining are consistent with the number of PIXAR. Therefore, to ensure the fairness of the comparison experiment, it is necessary to construct a multilingual dataset with a similar dataset size. Therefore, this project chose the same Wikipedia dataset used in PIXAR and selected the first 1024,000 samples in each language [4] [7].

The first column of table 3.2 demonstrates the configuration of rendering the raw text to images. For most of the parameters, keep them the same as those in the original PIXAR paper. In terms of patch size, this project chooses to try a larger patch size: 16 * 16. When the font size is unchanged, a larger patch size will increase the sequence length contained in each patch, and it will also increase the difficulty of image generation.

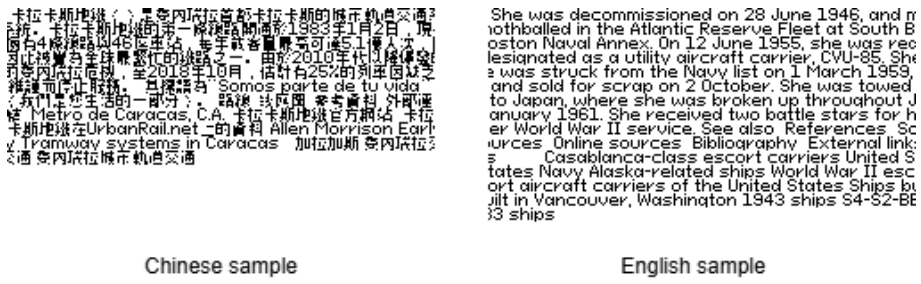


Figure 4.1: This image shows the comparison of Chinese image and English image

4.2 Computational budget

Stage 1 training of PIXAR++ models is trained in the same environment of PIXAR [7]. They both completed around 90 hours through 16 NVIDIA V100 GPUs. Stage 2 training and evaluation for PIXAR++ are also in the same environment as PIXAR. Because the training resources required were much less than stage 1, 4 NVIDIA V100 GPUs were used for stage 2 training and evaluation.

4.3 Pretraining stage

In the pretraining phase of this project, two different datasets of rendered images were used as inputs. The size of the input images is different in each of the models. The input data of the first model was the images with the size of 240 * 192, which is made up of 720 patches and each patch size is 8 * 8. The input image size of the second model is 480 * 384, which consists of 720 patches and each patch size is 16 * 16. The other difference is that when training a model with an input patch size of 8 * 8, the dataset was rendered before training began. When training the model with a patch size of 16*16, the parallel mode of CPU and GPU was used. The CPU rendered the data, while the GPU trained the model. Based on the training time, the second type of training did not increase much training time. In addition, the size of the first method dataset is 1.2T, while the size of the second method dataset is 28G.

To control variables, the patch size was increased without changing the font size. The main purpose of increasing patch size is to increase the number of pixels generated each time, and then increase the length of the generated sequence with a fixed font size. According to the training results, the training loss of the PIXAR++ using the small image as input is 0.18, while the training loss of the PIXAR++ using the large image as input is 0.14. However, in downstream experiments, the PIXAR++ using the larger

picture performed worse than the other PIXAR++. Because the training uses binary images and the loss function is the usual pixel-wise binary cross-entropy loss, the white part of the large image is much more than the white part of the small image. Therefore, in the case of the same font size, the line spacing in the large picture is larger, and the probability of this part being predicted correctly is higher. As a result, models using larger images have less training loss.

4.4 GAN stage

Based on the experiments mentioned by Yintao et al., the best-performing model was found at step 200. However, the original evaluation frequencies of the training are 200. Therefore, this project chose to reduce the total training steps to reduce the training time. In addition, the evaluation steps were reduced to 100 or 50 to get more checkpoints around 200 steps [7]. The chosen checkpoints of PIXAR++$_{stage2}$ are 100 steps 8-patch-size checkpoint and 300 steps 8-patch-size checkpoint.

4.5 Discriminative Tasks

4.5.1 GLUE

Based on the paper of Tai et al. and Rust et al. [4] [7], the GLUE benchmark was chosen as the primary metric to test the language understanding of the model. GLUE contains 1 regression and 8 classification tasks. A newly created prediction head from the rendered data is used to finetune PIXAR++ and the rendering of the dataset follows the approach used for the training dataset. Some tasks consist of a pair of sentences, and to separate the two sentences, a black patch is inserted between them. The embedding of the last black patch is used as the head of the task input which is the same as the paper of PIXAR. All hyperparameters are the same as those in the paper of PIXAR. Besides, the early-stopping strategy mentioned in the paper on PIXAR is also used in the experiments of this project [7].

PIXAR++$_{stage1}$

According to the experiment result for stage one, the PIXAR++$_{stage1}$ models with the same input patch size as PIXAR$_{stage1}$ perform slightly worse than PIXAR$_{stage1}$ models, given the same parameters of their models. This thesis will compare these two models

first. For the average value of all tasks, PIXAR$_{stage1}$ is 74.0, and PIXAR++$_{stage1}$ with 8 patch size is 71.3, which is only 2.7 lower. This may be because the GLUE benchmark uses English as the dataset language, PIXAR$_{stage1}$ uses 26.8M English samples as the training dataset, but only about 1/7 of the 27.1M samples in PIXAR++$_{stage1}$ are in English. Therefore, PIXAR++$_{stage1}$ should not perform as well as PIXAR$_{stage1}$ on GLUE. Another reason may be that other languages also have a disturbing effect on the parameters of the model. Compared to GPT-2 [44] and BERT [55], the two PIXAR++$_{stage1}$ models outperformed GPT-2 on STSTB, MRPC, RTE, and WNLI and outperformed BERT on WNLI. This suggests that PIXAR++$_{stage1}$ performs better on tasks with smaller datasets.

Specifically, for the single-sentence tasks in GLUE (CoLA and SST-2), 8-patch-size PIXAR++$_{stage1}$ performs worse than PIXAR$_{stage1}$. For the accuracy of the SST-2 task, 8-patch-size PIXAR++$_{stage1}$ is close to PIXAR$_{stage1}$, while Matthew's correlation of the CoLA task has a large gap between these two models. The slight gap on SST-2 can be interpreted as a difference in the dataset. For CoLA, the primary sources of its dataset are books and articles. The pretraining dataset of PIXAR$_{stage1}$, however, contains Bookcorpus [42] and is, therefore, better suited to this task. In addition, since the task is to determine whether the syntax is correct. The PIXAR++$_{stage1}$ dataset contains seven languages, so the syntax of languages other than English can affect the judgment of the model. For the similarity and paraphrase tasks (MRPC, QQP, and STS-B), the 8-patch-size PIXAR++$_{stage1}$ all have good performance and are close to PIXAR$_{stage1}$ scores. For the inference tasks (MNLI, QNLI, RTE, and WNLI), the 8-patch-size PIXAR++$_{stage1}$ still achieved good performance. For RTE and WNLI, the performance of 8-patch-size PIXAR++$_{stage1}$ is even better than that of PIXAR$_{stage1}$. For RTE, the reason could be that the data of this task are from a Wikipedia dataset, and PIXAR++$_{stage1}$ only used the Wikipedia dataset in pretraining. For WNLI, The reason may be that the dataset of WNLI is too small and unstable.

However, 16-patch-size PIXAR++$_{stage1}$ performed worse than the other two models on each task of GLUE. Therefore, this project will not analyze the reasons behind each task individually. There are many reasons for this problem. The first reason is to increase the size of the input image and the size of the patch that needs to be predicted. Because of this change, the number of pixels in the input picture has increased, and at the same time, the number of pixels in the patch that needs to be predicted has also increased. This means that the number of features the model needs to learn has increased, however, in the model, the overall number of parameters has not changed, so

Models	Parameters	Patch size (pixel)	MNLI-m/mm 392k	QQP 363k	QNLI 108k	SST-2 67k	COLA 8.5k	STSB 5.7k	MRPC 3.5k	RTE 2.5k	WNLI 635	AVG
GPT-2	126M	NA	81.0	89.4	87.7	92.5	77.0	74.9	71.5	52.0	54.9	75.6
BERT	110M	NA	84.0/84.2	87.6	91.0	92.6	60.3	88.8	90.2	69.5	51.8	80.0
PIXAR$_{stage1}$	85M	8	78.4 / 78.6	85.6	85.7	89.0	39.9	81.7	83.3	58.5	59.2	74.0
PIXAR++$_{stage1}$	85M	8	75.2 / 75.4	84.4	83.6	87.7	21.7	80.9	82.4	**60.6**	**60.6**	71.3
PIXAR++$_{stage1}$	85M	16	70.0 / 70.2	83.2	82.2	83.5	10.9	77.2	81.8	57.0	57.7	67.4
PIXAR$_{stage2}$	85M	8	79.7 / 80.1	86.3	85.7	89.3	37.0	82.4	82.8	57.7	60.6	74.2
PIXAR++$_{stage2}$	85M	8	74.5 / 75.2	84.4	83.6	86.9	15.9	80.3	81.8	**62.1**	57.7	70.3
PIXAR++$_{stage2}$	85M	16	71.2 / 71.4	83.0	82.6	83.9	10.0	76.4	82.7	58.8	56.3	67.6

Table 4.1: This table shows the performance of BERT [55], GPT2 [44], PIXAR and PIXAR++. PIXAR++ achieves similar performance to PXIAR on GLUE. For QQP and MRPC, the F1 score is used as the benchmark. The Spearman's ρ is used for STSB and Matthew's correlation is applied for COLA. The accuracy is used on other tasks.

the evaluation results are not good. Another reason is that the training stage uses the usual pixel-wise binary cross entropy loss as the loss function, but in this loss function, the weight of the white and black pixels is the same. However, in the large patch size, due to the same size, the white pixel occupies a larger proportion, and the white pixel is easier to predict correctly than the black pixel. Therefore, although the loss value during pretraining is low, the effect of evaluation is not good. A good solution is to balance the weight of the two types of pixels or increase the weight of the black pixel. Another solution is to increase the font size of the large image, however, this will reduce the length of the predicted sequence.

PIXAR++$_{stage2}$

The last three rows in the table 4.1 show the performance of the stage 2 models on the GLUE benchmark. In terms of average performance, all models achieved similar results at stage 1 and stage 2. Specifically, the average score of PIXAR$_{stage2}$ and 16-patch-size PIXAR++$_{stage2}$ has a small increase, while the average score of 8-patch-size PIXAR++$_{stage2}$ has a small decrease. The reason may be that stage 2 models are trained to increase the readability and accuracy of the generated text to improve their ability to handle generative tasks. Therefore, the development set used to select stage 2 PIXAR++ checkpoints is a validation set from LAMBADA. As a result, the checkpoints picked out may not perform best on discriminative Tasks. Besides, the performance of PIXAR++$_{stage2}$ models on RTE is still better than PIXAR$_{stage2}$ for the same reason mentioned in the PIXAR++$_{stage1}$ section. The surprising result was that 16-patch-size PIXAR++$_{stage2}$ performed better than 8-patch-size PIXAR++$_{stage2}$ on MRPC tasks

and is very close to the performance of PIXAR$_{stage2}$. The difference between the two PIXAR++ models is only patch size. Therefore, since the font size of these two models is the same and the training of PIXAR++ is pixel-based, during training, the patch of the input and output images of 16-patch-size PIXAR++ contain more characters and information. This difference may be why 16-patch-size PIXAR++$_{stage2}$ performs better when dealing with online news datasets.

Hyperparameters

Table 4.2 shows the hyperparameters chosen for the evaluation of the GLUE benchmark. All parameters are selected based on the description of the PIXAR paper. For larger tasks like MNLI and QQP, select 256 as the batch size and 8000 as the max steps. For smaller tasks like STSB and WNLI, 32 is selected as the batch size, while 2000 and 500 are selected as the max steps.

PIXAR++$_{stage1}$	MNLI	QQP	QNLI	SST-2	COLA	STSB	MRPC	RTE	WNLI
lr	3e-5	3e-5	3e-5	3e-5	3e-5	3e-5	6e-5	3e-5	3e-5
Weight decay	0.1	0.1	0.1	0.01	0.01	0.01	0.01	0.01	0.01
Optimizer	AdamW								
Warmup	Linear warmup								
Warmup steps	1000	1000	500	200	50	100	20	50	2
β_1	0.9								
β_2	0.95								
Random seed	42								
Batch size	256	256	256	256	256	32	64	32	128
Max steps	8000	8000	4000	2000	500	2000	500	500	20
evaluation freq.	500	500	200	200	100	100	50	50	Ĩ epoch
PIXAR++$_{stage2}$	MNLI	QQP	QNLI	SST-2	COLA	STSB	MRPC	RTE	WNLI
lr	3e-5	3e-5	3e-5	3e-5	3e-5	3e-5	6e-5	3e-5	3e-5
Weight decay	0.1	0.1	0.1	0.01	0.01	0.01	0.01	0.01	0.01
Optimizer	AdamW								
Warmup	Linear warmup								
Warmup steps	1000	1000	500	200	50	100	20	50	2
β_1	0.9								
β_2	0.95								
Random seed	42								
Batch size	256	256	256	256	256	32	64	32	128
Max steps	8000	8000	4000	2000	500	2000	500	500	20
evaluation freq.	500	500	200	200	100	100	50	50	Ĩ epoch

Table 4.2: This table shows the hyperparameters applied in the GLUE benchmark.

4.5.2 XNLI

XNLI is the downstream task translated from MNLI in the GLUE benchmark [12]. This original dataset had no index, which led to poor training results. However, there is an index item in the MNLI dataset. Therefore, in the preprocessing stage, an index item is added to each sample in the dataset dictionary. Twelve languages in the XNLI task are chosen as the experiment languages. Six of these languages were the same as those in the pretraining dataset which are Arabic, German, English, Spanish, French, and Russian. Italian was not included as the experimental language because it was not included in the XNLI dataset. Table 4.3 shows the performance of these languages on BiLSTM-max, BERT, PIXAR, and PIXAR++. Six other languages including Bulgarian, Greek, Urdu, Swahili, Turkish, and Vietnamese were chosen to test the performance of PIXAR++ models in languages they had not been trained before. Table 4.4 shows the performance of these languages on the same models of table 4.3. The results of the BiLSTM-max are provided from the paper of XNLI [12].

Models	Parameters	Patch size	ar	de	en	es	fr	ru	AVG
BiLSTM-max	NA	NA	65.8	66.5	73.7	68.8	68.3	66.5	68.2
BERT	110M	NA	70.7	75.9	81.9	77.8	NA	NA	NA
PIXAR++$_{stage1}$	85M	8	63.6	68.0	75.3	72.1	71.2	67.0	69.5
PIXAR++$_{stage1}$	85M	16	55.2	66.3	70.9	70.0	68.0	65.1	65.9
PIXAR$_{stage2}$	85M	8	59.7	67.2	78.8	69.8	67.7	64.0	67.9
PIXAR++$_{stage2}$	85M	8	62.6	70.0	74.7	72.8	71.2	68.6	70.0
PIXAR++$_{stage2}$	85M	16	54.1	66.9	70.9	69.4	68.3	64.5	65.7

Table 4.3: This table shows the performance of BiLSTM-max, BERT, PIXAR, and PIXAR++ models on XNLI tasks. The metric used here is accuracy. The XNLI was translated from MNLI and is used to evaluate the performance of multilingual models. This table mainly shows the performance of the languages present in the pretraining dataset of the PIXAR++. These languages are ar(Arabic), de(German), en(English), es(Spanish), fr(French), and ru(Russian).

PIXAR++$_{stage1}$

Languages in the pretraining dataset: According to the table 4.3, the PIXAR++$_{stage1}$ model using 8 patch size images as input outperformed the other PIXAR++$_{stage1}$ model overall. In addition, PIXAR++$_{stage1}$ outperforms BiLSTM-max in all languages except Arabic. The accuracy of BiLSTM-max on Arabic is 65.8, but on the 8-patch-size and

Models	Parameters	Patch size	bg	el	ur	sw	tr	vi	AVG
BiLSTM-max	NA	NA	67.4	66.4	56.6	58.2	64.5	66.0	63.1
BERT	110M	NA	NA	NA	61.6	NA	NA	NA	NA
PIXAR++$_{stage1}$	85M	8	67.8	68.0	54.3	60.5	64.7	63.9	63.2
PIXAR++$_{stage1}$	85M	16	64.7	64.5	50.3	58.2	61.6	61.3	60.1
PIXAR$_{stage2}$	85M	8	60.5	64.6	50.2	56.2	65.5	63.9	60.2
PIXAR++$_{stage2}$	85M	8	67.8	67.0	55.7	61.4	64.4	64.4	63.5
PIXAR++$_{stage2}$	85M	16	64.5	64.5	50.4	58.5	61.0	61.7	60.1

Table 4.4: This table shows the performance of BiLSTM-max, PIXAR, and PIXAR++ models on other XNLI tasks. The metric used here is accuracy. This table mainly shows the performance of the languages that were not included in the pertaining dataset. These languages are bg(Bulgarian), el(Greek), ur(Urdu) (low-resource language), sw(Swahili), tr(Turkish), and vi(Vietnamese).

16-patch-size PIXAR++$_{stage1}$, they are 63.6 and 55.2. The reason may be that in the training language of the PIXAR++ model, Arabic is very different from the other six languages. However, the other six languages are similar. Therefore, during the stage one training of PIXAR++$_{stage1}$ on Arabic samples, other languages in the pretraining dataset can not provide useful features and even have negative effects. Despite this, the result of the Arabic task is also close to the result of BiLSTM-max. For the 8-patch-size and 16-patch-size PIXAR++$_{stage1}$, the prediction accuracy of these two models for English tasks is the highest, which is 75.3 and 70.9 respectively. The reason may be that the total number of characters in the English samples in the pretraining datasets is more than the number of other languages. As a result, the English samples in the preprocessed pretraining datasets are more likely to be longer and more than other languages.

Languages did not in the pretraining dataset: According to the table 4.4, the 8-patch-size PIXAR++$_{stage1}$ still performs better than the other PIXAR++$_{stage1}$ overall. The average performance of PIXAR++$_{stage1}$ is better than the BiLSTM-max. This shows that the PIXAR++$_{stage1}$ model has good processing ability even when faced with language text that has never been seen before. Besides, a low-resource language, Urdu, is also one of the test languages. The task for this language is very challenging and as expected, PIXAR++$_{stage1}$ performs the worst on the task of this language. The accuracy of the two PIXAR++$_{stage1}$ is only 54.3 and 50.3. However, the performance of 8-patch-size PIXAR++$_{stage1}$ is still comparable to the baseline model which is 56.6. This shows that PIXAR++$_{stage1}$ can still perform well even in the face of a low-resource language that has never been seen before. All four languages except Vietnamese and

Urdu exceeded the baseline in accuracy. This shows that the model can handle languages not existing in pretraining.

PIXAR++$_{stage2}$

Languages in the pretraining dataset: The last three rows of the table 4.3 show the performance of the stage 2 PIXAR and PIXAR++. On average, 8-patch-size PIXAR++$_{stage2}$ slightly outperforms PIXAR++$_{stage1}$, while the average score of 16-patch-size PIXAR++$_{stage2}$ is slightly lower than PIXAR++$_{stage1}$. Besides, the average accuracy of 8-patch-size PIXAR++$_{stage2}$ is higher than it of the PIXAR$_{stage2}$ and 8-patch-size PIXAR++$_{stage2}$ performed better than PIXAR$_{stage2}$ in all five languages except English. The accuracy of PIXAR$_{stage2}$ and 8-patch-size PIXAR++$_{stage2}$ in English is 78.8 and 74.7. The reason concerns the datasets for pretraining and stage 2 training, PIXAR uses a pure English training dataset, while the dataset of PIXAR++ contains seven languages. In addition, the training dataset of PIXAR has 26M English samples, while PIXAR++ has only about 1/7 English samples in its training dataset. Since the 16-patch-size PIXAR++$_{stage1}$ did not perform as well as the other PIXAR++$_{stage1}$ on stage 1, within the expectation, its performance after stage 2 training is still lower than the other PIXAR++$_{stage2}$.

Languages did not in the pretraining dataset: The last three rows of the table 4.4 show the performance of the stage 2 PIXAR and PIXAR++. Based on the average accuracy of these languages of the three models, 8-patch-size PIXAR++$_{stage2}$ got 63.5 which is higher than the other two models and the average accuracy of the other two models is similar, which is 60.2 for PIXAR$_{stage2}$ and 60.1 for PIXAR++$_{stage2}$. 8-patch-size PIXAR++$_{stage2}$ also slightly outperforms PIXAR++$_{stage1}$, while the average score of 16-patch-size PIXAR++$_{stage2}$ is same as it of PIXAR++$_{stage1}$. This may be because the PIXAR++ training dataset contains more letter types, grammar, and syntactic formats. Therefore, PIXAR++ is more robust when facing unknown languages. Specifically, 8-patch-size PIXAR++$_{stage2}$ outperforms PIXAR$_{stage2}$ in all languages except Turkish. The reason may be that Turkish contains similar letters and words to English.

Hyperparameters

The hyperparameters for this task are the same as MNLI tasks in the GLUE benchmark. This is because other language datasets of XNLI are the translation version of MNLI.

Models	lr	Weight decay	Optimizer	Warmup	Warmup steps	β_1	β_2	Random seed	Batch size	Max steps	evaluation freq.
PIXAR++$_{stage1}$	3e-5	0.1	AdamW	Linear warmup	1000	0.9	0.95	42	256	8000	500
PIXAR++$_{stage2}$	3e-5	0.1	AdamW	Linear warmup	1000	0.9	0.95	42	256	8000	500

Table 4.5: This table shows the hyperparameters used in XNLI downstream tasks.

4.6 Generative tasks

In the experiments of generative tasks, the prompt was rendered as images, and a white patch of 3 pixels in length was inserted before the generation began. The white patch is used as a space to separate new words. PIXAR++ generates new text image patches autoregressively from here [7]. This project mainly selects bAbI and LAMBADA for the generative tasks. The bAbI only has an English version but the LAMBADA tasks have English, French, German, Italian, and Spanish versions [44]. These two generative tasks were used in PIXAR papers. The bAbI task is a QA task that evaluates the reading comprehension of the model in providing the truth. The prompt is designed to contain four examples from bAbI and uses "|" as the divider between the question and the answer. LAMBADA is the benchmark used to test the text-understanding ability of LLMs. The model needs to provide a prediction for the last word of a sentence after reading a paragraph [7].

Table 4.6 shows the results of PIXAR and PIXAR++ models on bAbI and LAMBADA tasks. Since the training set of PIXAR includes only English, the performance results of PIXAR in the tasks of other languages are labeled NA in this table. In addition, all LAMBADA tasks tested with PIXAR++ have three metrics. The one on the left is the readability for a single language. Specifically, the meaning of it is whether the generated text is in the same language vocabulary as the prompt. In the middle is the readability of the five languages, which tests whether the generated text is in any of the vocabularies of the five LAMBADA tasks in different languages. The last value is the accuracy of the predicted result. In addition, for readability, a portion of all generated text is not in the same language as the prompt but is still readable. The reason may be that multiple languages are used in the pretraining stage. Because the words of some languages are relatively similar, the model may misjudge the language of the text to generate when performing the generation task. Besides, since the dataset used for PIXAR contains only English, the performance of PIXAR on the bAbI task and the English LAMBADA task should be more advantageous.

PIXAR++$_{stage1}$

According to the experimental results, in the model of stage one, PIXAR has the highest prediction accuracy in bAbI and English LAMBADA tasks. Since PIXAR only used the English dataset and was trained on more English samples than PIXAR++, more English content information, sentence structure, grammar, and words were learned by PIXAR. In addition, because pure English datasets use all English characters, they are not affected by the noise and perturbation generated by characters contained in other languages when generating text. It is worth noting that 8-patch-size PIXAR++$_{stage1}$ performs better than PIXAR$_{stage1}$ in readability. This shows that using multilingual data sets can increase the robustness of generating sequences as readable text. In addition, 8-patch-size PIXAR++$_{stage1}$ generates much more accurate text on LAMBADA tasks and bAbI tasks in all languages than 16-patch-size PIXAR++$_{stage1}$. In terms of readability, 8-patch-size PIXAR++$_{stage1}$ is also better than 16-patch-size PIXAR++$_{stage1}$, except for the readability of the five languages of Italian. This may be because 16-patch-size PIXAR++$_{stage1}$ is more pixels than 8-patch-size PIXAR++$_{stage1}$ in the patch size of the input image during training and each generated patch size. As a result, larger models and longer time may be required to train 16-patch-size PIXAR++$_{stage1}$.

PIXAR++$_{stage2}$

According to the experimental results, the readability of PIXAR on the bAbI task improved from 63.2 to 77.0 (Growth value: 13.8), and on the English LAMBADA task, it improved from 54.8 to 82.2 (Growth value: 27.4). For 8-patch-size PIXAR++$_{stage2}$, the Growth values are 0.4 and 0.5 and for 16-patch-size PIXAR++$_{stage2}$, the growth values are - 0.3 and 0.2. This means GAN loss is more useful for PIXAR which uses a single language as the training dataset than PIXAR++ which uses a multi-language dataset. Since the best checkpoints for PIXAR++$_{stage2}$ are around 200 steps and the batch size is 32, only a small fraction of the multilingual datasets are used and the number of samples in each language is unbalanced. Also, since GANs are very unstable, the checkpoints used for these experiments may not be the best. Finally, since the samples of the multilingual dataset contain more words, characters, grammatical structures, and syntactic structures, this dataset was more difficult to train. Given these factors, the improvement in the accuracy of PIXAR$_{stage2}$ generated text is still larger than that of PIXAR++$_{stage2}$. For PIXAR$_{stage2}$, the improved values of bAbI and LAMBADA are 8.5 and 8.1. But for 8-patch-size PIXAR++$_{stage2}$ they are 3.4 and

1.1 and for 16-patch-size PIXAR++$_{stage2}$, they are 0.4 and 0.3. However, even so, the accuracy of all prediction texts improved after stage 2 training. This shows that using GAN loss as the final layer of the model can also improve the performance of PIXAR++ in generative tasks.

Models	Parameters	Patch height (pixels)	bAbI	LAMBADA (en)	LAMBADA (de)	LAMBADA (es)	LAMBADA (fr)	LAMBADA (it)
PIXAR$_{stage1}$	113M	8	63.2 (11.1)	54.8 (5.7)	NA	NA	NA	NA
PIXAR++$_{stage1}$	85M	8	61.0 (9.7)	**61.9** / 63.4 (1.9)	45.2 / 57.3 (2.8)	50.1 / 56.7 (0.9)	47.0 / 53.6 (3.5)	50.6 / 57.1 (2.6)
PIXAR++$_{stage1}$	85M	16	42.3 (4.6)	54.3 / 55.3 (0.5)	35.5 / 44.3 (1.2)	39.8 / 48.5 (0.2)	38.8 / 43.0 (0.9)	47.5 / 57.4 (0.9)
PIXAR$_{stage2}$	113M	8	77.0 (19.6)	82.2 (13.8)	NA	NA	NA	NA
PIXAR++$_{stage2}$	85M	8	61.4 (13.0)	66.4 / 68.2 (3.0)	49.8 / 61.3 (3.2)	53.9 / 61.0 (1.5)	51.0 / 56.5 (3.8)	55.6 / 63.1 (4.2)
PIXAR++$_{stage2}$	85M	16	42.0 (5.0)	54.5 / 55.5 (0.8)	37.5 / 46.9 (1.2)	40.7 / 48.4 (0.3)	37.0 / 41.8 (1.2)	50.7 / 55.9 (1.1)

Table 4.6: This table shows the performance of PIXAR and PIXAR++ on two generative tasks LAMBADA and bAbI. Among them, the performance of these models on the bAbI task is presented by the readability ratio and the few shot accuracy (in brackets). For LAMBADA is the readability ratio for one language, readability ratio for 5 languages, and zero-shot last-word prediction accuracy (in brackets).

Output analysis

Figure A.1 and A.2 show the good and bad examples of LAMBADA generated by 8-patch-size PIXAR++. These examples of A.1 generate the wrong text for the following reasons: (1) According to the prompt, ”the man smiled at him.” was in the first line and this ”him” represents ”carlos”. The generated text is ”him.” which means the model did not understand what ”him” is, but only copied the answer from the prompt. (2) The reason for this German example is similar. Besides, the prompt of this example did not have the same word or a word with similar meaning as the result. The meaning of the word ”Looks” in English is similar to ”ansah”, however, ”Look” means ”make people” in German. Therefore, this is a tough sample to predict. (3) This example does not predict correctly because the answer is ”Shane” but this word appears at the beginning of a sentence in the prompt, which means no useful information in front of ”Shane” but only a period. Besides, the last word in the prompt, ”cuenta,” doesn’t appear in the previous paragraph either. (4) The predicted text of this example is meaningless. The reason could be the model did not find a similar phrase or a proper word from the prompt. (5) The answer here is “combattimenti”, a synonym of ”lotta.” However, due to the phrase structure before the result, there is no equivalent in the prompt. So the model doesn’t even answer ”lotta.” In Figure A.2, all result predictions are correct because the phrase containing the result has appeared in the previous prompt. For example, ”the

old city of suzhou" appeared in the previous prompt since the last word of this prompt is old, the model outputs the following words "city of suzho" in this phrase. Since the output length is limited, the last letter of the word "suzhou" was not generated.

Figure A.3 and A.4 show the good and bad examples of LAMBADA generated by 16-patch-size PIXAR++. The reasons for the wrong samples are: (1) The reason for the first example has been mentioned before, which is "no similar phrase" in the previous prompt. Besides, since the answer is "cooking", although the model finds the word "cook" as the answer, it is difficult for it to change this word to "cooking". (2) The result for this answer was not provided in the previous prompt, which makes this sample difficult to predict. (3) Since in the prompt, the symbol ":" was after the word "dijo", the result the model predicted is ":". (4) There are two phrases "de la pousser" and "de la jeep" in the prompt. The predicted text is similar to the first three characters of "pousser". The reason could be the model thinks the word "pousser" is more likely to be the answer. (5) This is the same sample mentioned in the 8-patch-size examples. The reason for the failure is the same, and the purpose of showing it is to compare it with the image generated by 8-patch-size PIXAR++. The reason for the correct prediction is the same as that of the 8-patch-size PIXAR++.

Figure A.5 and A.6 show the good and bad examples of bAbI. According to figure A.5, PIXAR++ makes the mistake because it only learns a fixed structure "Where is ...? | " but not the meaning of the sentences. Figure 4.2 shows where the model found the answer visually, the answer to the upper prompt is "office" and the other is "garden". According to this figure, the result that PIXAR++ generated is according to the answer to the same question in the previous prompt but not the last place "Sandra" went. Figure A.6 provides another reason for the model to make incorrect predictions, which is the misspelling of words. For example, the letter "g" in the generated word "garden" was more like "a" and the letter "a" in the generated word "bearoom" should be "d". This may be because of the generated noises in the prediction period.

Figure 4.2: This image shows where the model found the answer

Chapter 5

Conclusion and Discussion

5.1 Conclusion

Achievements: This project proposed PIXAR++, the extended version of PIXAR. PIXAR is the first pixel-based autoregressive LLM that can generate images of a short text sequence [7]. However, the pretraining dataset of PIXAR is only based on English and the patch size of the input and output images is fixed to 8 * 8. Therefore, a 7 language dataset was collected and created to train PIXAR++ and 8 * 8 and 16 * 16 patch sizes are tried in this project. Under the premise of the same font size, a larger patch will contain more text sequences. Besides, some downstream tasks are used to test the performance of PIXAR++. GLUE and XNLI are used to show the performance of PIXAR++ on discriminative tasks. Since GLUE is a pure English benchmark, PIXAR performs better, but 8-patch-size PIXAR++ performs similarly to PIXAR. Since XNLI is a multilingual task, 8-patch-size PIXAR++ outperforms PIXAR in most languages. In addition, 16-patch-size PIXAR++ performs worse than the other two models on both discriminative tasks. For the free-text QA generation tasks, the project chose the bAbI task and 5 language LAMBADA tasks. In the English generation task, PIXAR performed better than the other two models. PIXAR did not experiment with LAMBADA in other languages. Therefore, for tasks in other languages, there are only two PIXAR++ experimental results. For all the generative tasks, 8-patch-size PIXAR++ still performs better than 16-patch-size PIXAR++.

Limitations and future work : Due to the difference in the number of white and black pixels in the patches, although the model using a larger patch size has a lower training loss, its performance in the downstream task is no better than 8-patch-size PIXAR++. Therefore, Balanced Cross-Entropy and Focal loss is a better choice of loss

function. In addition, due to the increase in language types and patch size, the size and training time of the model selected in this project may be insufficient. Therefore, larger models and longer training times could be used in future work. Besides, based on the experimental results of the generative task, PIXAR++ finds the answer by looking for whether the last several letters in the prompt were present in the previous prompt. If present, the text sequence following these letters in the previous prompt is generated; if not, there is no way to predict correctly. This shows that the model has not learned the correct dependencies between texts over long distances and that the dependencies between patches are poorly interpretable. In addition, larger datasets, more languages, Larger font sizes, and Higher resolution ratios could be tried in the future, if the computational resources are sufficient. Finally, although the GAN model improves the performance in generative tasks, it is still unstable and the diffusion model proves to perform better than the GAN model on Image Synthesis [8]. Therefore, using the diffusion model in stage two is worth trying. The details of the future work are in the discussion section.

In summary, the project expanded PIXAR to handle more languages which proves the possibility of learning text information from pixels in other languages, and expanded the patch size to increase the length of the generated text. The experiment of this project extends the application scope of PIXAR++ and provides more possibilities for the extension of pixel-based models.

5.2 Discussion

Balanced Cross Entropy & Focal Loss

In this project, the binary cross entropy (CE) loss is chosen as the loss function in the training of PIXAR++ [56]. However, the number of white pixels and the number of black pixels per patch of the input images and the generated patches are not equal. In addition, in images with 16 * 16 pixels per patch, the two classes are more unbalanced due to the increase in line spacing and the increase in white space after the end of the text. Therefore, if their weights are the same, it will result in poor training results even though the loss function value is small. The equation of the binary cross entropy is:

$$CE(p,y) = \begin{cases} -\log(p) & \text{if} \quad (\text{y} = 1) \\ -\log(1-p) & \text{otherwise} \end{cases} \tag{5.1}$$

Where y means white or black pixel in this project and $p \in [0,1]$ means the estimated probability of the PIXAR++ model for y is the black pixel [56].

$$p_t = \begin{cases} p & \text{if } (\text{y} = 1) \\ 1-p & \text{otherwise} \end{cases} \tag{5.2}$$

Therefore, the CE loss can be written as this equation for convenience:

$CE(p,y) = CE(p_t) = -\log(p_t)$

However, this loss function can not solve the problem of class imbalance. Therefore, a common idea was proposed to use a weighting factor $\alpha \in [0,1]$, where α is for the black pixel class and 1 - α is for the white pixel class. The notation definition of α_t is same as p_t [56]. The equation of α-balanced CE loss equation is:

$CE(p_t) = -\alpha_t \log(p_t)$

Although the importance of white and black samples was balanced by α-balanced CE loss, the easy and hard samples are not distinguished. Therefore, the focal loss (FL) loss function was designed to reduce the weight of the easy examples. The focal loss function is:

$FL(p_t) = -(1-p_t)^{\gamma}\log(p_t)$

Where γ is a tunable hyperparameter between $[0,5]$. In the experiment of the paper on FL, FL loss works the best with $\gamma = 2$.

Larger models & longer training time

According to the paper of PIXAR, the PIXAR model with 113M parameters was chosen to deal with the generative tasks [7]. However, the PIXAR++ has only 85M parameters. Besides, since the dataset used on PIXAR++ is a multi-language dataset, more characters, words, syntactic structures, and grammar need to be learned by the model, which will need more parameters. In addition, this project attempts to image with 16 * 16 pixels per patch as the input and output of training. Therefore, each patch contains more information and is more difficult to train. Moreover, compared with 8 * 8 pixels per patch, the generated patch is larger and contains longer text length, which makes it more difficult to generate patches. Therefore, larger models and longer training times are necessary.

Lager dataset & more languages

Due to the limited training resources, only 85M models and 27M samples of seven language datasets were trained in this project. However, the English dataset for training PIXAR has 26M English samples [7]. Therefore, for multilingual datasets, to achieve experimental results similar to PIXAR on English tasks, there must be a similar number of samples in all languages which means at least 26M (samples per language) * 7 (Number of languages) samples used to train PIXAR++. In addition, as the learning difficulty of multiple languages is higher, the training difficulty will be higher due to the differences in characters, words, syntactic structure, and grammar between different languages. Therefore, samples for each language should be larger than 26M for good performance. Besides, the main reason that the PIXAR++ did not achieve good performance in Arabic is because other languages in the dataset are very different from Arabic. Therefore, more languages that are similar to Arabic should be added to the datasets to improve the ability to process Arabic tasks of PIXAR++ and other languages can also be chosen to train in PIXAR++ to increase the robustness of the PIXAR++ model.

Larger font size & Higher resolution ratio

As shown in A.1, in the French example, the word ”approprié” was written as ”appropriė”. Besides, according to 3.2, in the Chinese image on the left, the generated Chinese characters are very vague. Therefore, to adapt to the more complex characters in the language, the image resolution and font size of the model input should be increased appropriately. However, such modifications also require larger models and longer training times.

Diffusion models and longer generation

As mentioned in the motivation part, the performance of diffusion models is always better than the GAN models on Image Synthesis [8], and since the GAN model is unstable, although the automatic GAN ratio balancing is used, the stage 2 training is still difficult to optimize [7]. Therefore, diffusion models or diffusion transformers, which are also generative models, can be used to replace GAN models in future work. Besides, due to the readability metric, PIXAR++ still can not generate long sentences and further experiments will check if diffusion models can increase the length of the generated readable text [7].

Bibliography

[1] Ada Wan. Fairness in representation for multilingual nlp: Insights from controlled experiments on conditional language modeling. In *International Conference on Learning Representations*, 2021.

[2] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*, 2023.

[3] Omri Keren, Tal Avinari, Reut Tsarfaty, and Omer Levy. Breaking character: Are subwords good enough for mrls after all? *arXiv preprint arXiv:2204.04748*, 2022.

[4] Phillip Rust, Jonas F Lotz, Emanuele Bugliarello, Elizabeth Salesky, Miryam de Lhoneux, and Desmond Elliott. Language modelling with pixels. *arXiv preprint arXiv:2207.06991*, 2022.

[5] Keith Rayner, Sarah J White, and SP Liversedge. Raeding wrods with jubmled lettres: There is a cost. 2006.

[6] Zijun Sun, Xiaoya Li, Xiaofei Sun, Yuxian Meng, Xiang Ao, Qing He, Fei Wu, and Jiwei Li. Chinesebert: Chinese pretraining enhanced by glyph and pinyin information. *arXiv preprint arXiv:2106.16038*, 2021.

[7] Yintao Tai, Xiyang Liao, Alessandro Suglia, and Antonio Vergari. Pixar: Auto-regressive language modeling in pixel space. *arXiv preprint arXiv:2401.03321*, 2024.

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[9] Enis Karaarslan and Ömer Aydın. Generate impressive videos with text instructions: A review of openai sora, stable diffusion, lumiere and comparable models. *Authorea Preprints*, 2024.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[11] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[12] Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*, 2018.

[13] Tianyu Gao, Zirui Wang, Adithya Bhaskar, and Danqi Chen. Improving language understanding from screenshots. *arXiv preprint arXiv:2402.14073*, 2024.

[14] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[15] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[16] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[17] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022.

[18] Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Ocr-free document understanding transformer. In *European Conference on Computer Vision*, pages 498–517. Springer, 2022.

[19] Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning*, pages 18893–18912. PMLR, 2023.

[20] Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. Learning character-level compositionality with visual features. *arXiv preprint arXiv:1704.04859*, 2017.

[21] Baohua Sun, Lin Yang, Patrick Dong, Wenhan Zhang, Jason Dong, and Charles Young. Super characters: A conversion from sentiment classification to image classification. *arXiv preprint arXiv:1810.07653*, 2018.

[22] Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. Glyce: Glyph-vectors for chinese character representations. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] Falcon Z Dai and Zheng Cai. Glyph-aware embedding of chinese characters. *arXiv preprint arXiv:1709.00028*, 2017.

[24] Elizabeth Salesky, David Etter, and Matt Post. Robust open-vocabulary translation from visual text representations. *arXiv preprint arXiv:2104.08211*, 2021.

[25] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[26] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[27] Elizabeth Salesky, Neha Verma, Philipp Koehn, and Matt Post. Multilingual pixel representations for translation and effective cross-lingual transfer. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13845–13861, 2023.

[28] Michael Tschannen, Basil Mustafa, and Neil Houlsby. Clippo: Image-and-language understanding from pixels only. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11006–11017, 2023.

[29] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[30] Junyi Li, Wayne Xin Zhao, Jianyun Nie, and Ji rong Wen. Glyphdiffusion: Text generation as image generation. *arXiv preprint arXiv:2304.12519*, 2023.

[31] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

[32] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[33] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[34] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[35] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[36] Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*, 2019.

[37] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[38] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[41] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[42] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

[43] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[44] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[45] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[46] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[47] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher 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, 2013.

[48] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*, 2005.

[49] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[50] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[51] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[52] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

[53] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

[54] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[55] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[56] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

Appendix A

First appendix

A.1 First section

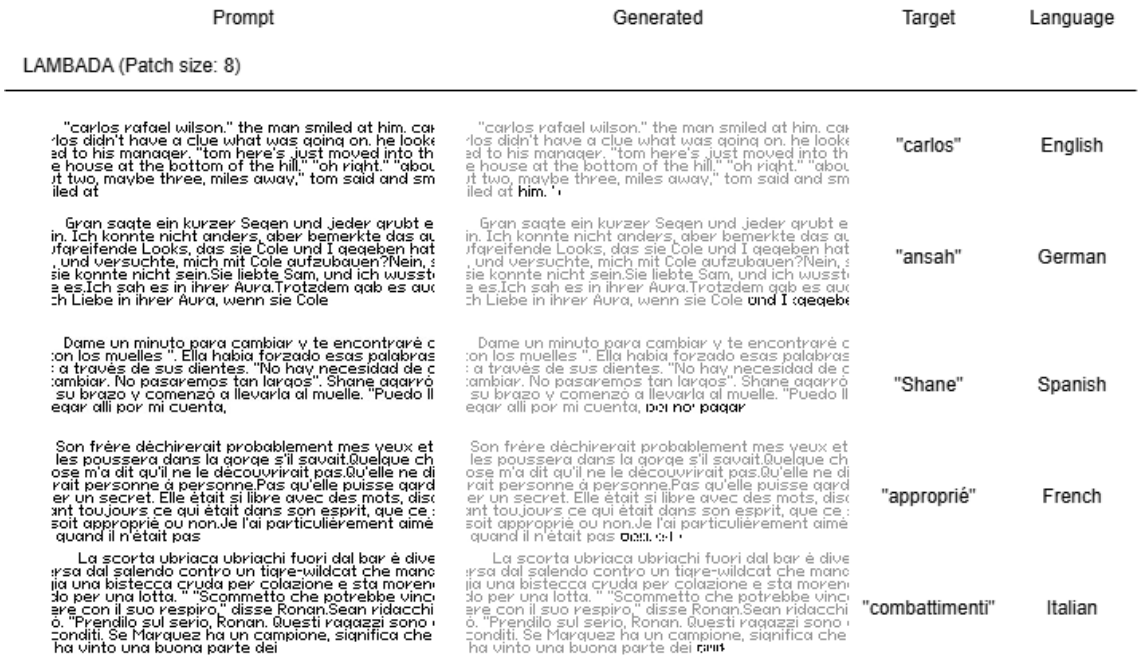


Figure A.1: This table shows some bad examples of the LAMBADA tasks for 8-patch-size PIXAR++$_{stage2}$

Figure A.2: This table shows some good examples of the LAMBADA tasks for 8-patch-size PIXAR++$_{stage2}$

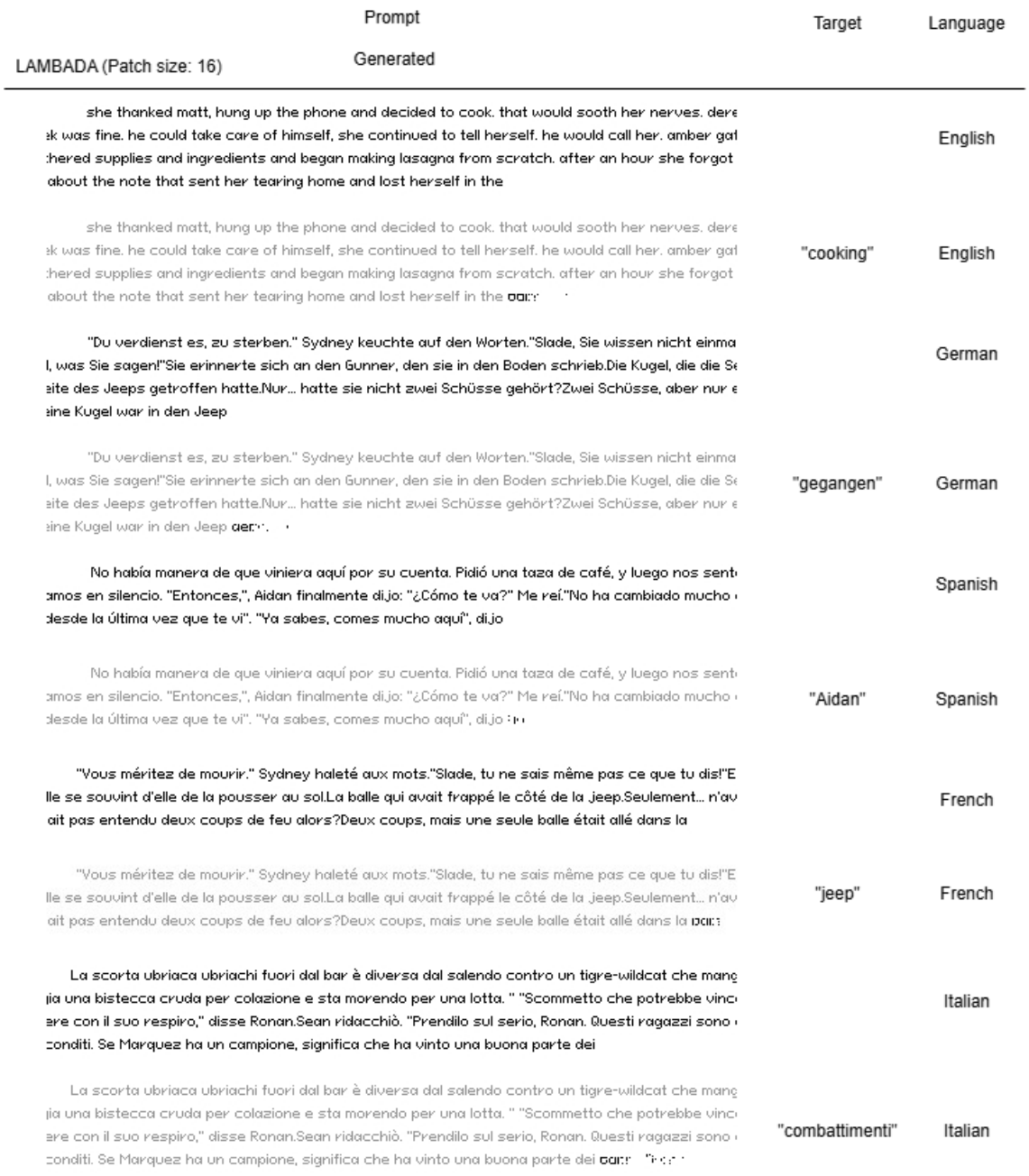


Figure A.3: This table shows some bad examples of the LAMBADA tasks for 16-patch-size PIXAR++$_{stage2}$

LAMBADA (Patch size: 16)	Prompt Generated	Target	Language
	to my surprise, the door was cracked open, and i could see two figures inside through the fros ted glass. i'd just raised my hand to knock when a voice drifted out to me. "but i don't want to be a healer," someone muttered. "i never wanted to be a		English
	to my surprise, the door was cracked open, and i could see two figures inside through the fros ted glass. i'd just raised my hand to knock when a voice drifted out to me. "but i don't want to be a healer," someone muttered. "i never wanted to be a healer "	"healer"	English
	"Ja, das wäre seltsam", sagte Gauner."Wir möchten vielleicht herausfinden, wer sonst über die Art von OCule-Manipulationen verfügt, die Professor Torret tut." "Außerhalb der Hochschule?" "Dra ußen oder in verschiedenen Disziplinen", sagte Gauner."Vielleicht sollte ich herausfinden, ob Profes sor Torret in denaphaos ein Gegenstück hat", sagte		German
	"Ja, das wäre seltsam", sagte Gauner."Wir möchten vielleicht herausfinden, wer sonst über die Art von OCule-Manipulationen verfügt, die Professor Torret tut." "Außerhalb der Hochschule?" "Dra ußen oder in verschiedenen Disziplinen", sagte Gauner."Vielleicht sollte ich herausfinden, ob Profes sor Torret in denaphaos ein Gegenstück hat", sagte Gauner "	"Gauner"	German
	"Entiendo eso, pero..." "Estamos navegando por Jacksonville a Norfolk. Podemos dejarte a lo prim ero en la mañana". "Necesito estar allí hace una hora". "¿Tienes una placa?" "Mi número de insignia es seis-cero-nueve-dos. No tengo el lujo-" "No tienes una		Spanish
	"Entiendo eso, pero..." "Estamos navegando por Jacksonville a Norfolk. Podemos dejarte a lo prim ero en la mañana". "Necesito estar allí hace una hora". "¿Tienes una placa?" "Mi número de insignia es seis-cero-nueve-dos. No tengo el lujo-" "No tienes una placa "	"placa"	Spanish
	"S'il te plaît, mes amis m'appellent Cameron et croyez-moi que vous préférez être mon ami qu e mon ennemi", a déclaré Cameron avec un rire. "Certainement, Cameron", dit Elijah se défilant la go rge et en relâchant son collier, "Que puis-je faire pour vous?" "N'oubliez pas que lorsque vous ave z donné une conférence sur" la différence entre le système juridique canadien et américain à l'Un iversité Queen's l'année dernière ", a déclaré		French
	"S'il te plaît, mes amis m'appellent Cameron et croyez-moi que vous préférez être mon ami qu e mon ennemi", a déclaré Cameron avec un rire. "Certainement, Cameron", dit Elijah se défilant la go rge et en relâchant son collier, "Que puis-je faire pour vous?" "N'oubliez pas que lorsque vous ave z donné une conférence sur" la différence entre le système juridique canadien et américain à l'Un iversité Queen's l'année dernière ", a déclaré Cameron auer	"Cameron"	French
	Vedendo questo piccolo magazzino di blocchi di informazioni antiche, post-moderni e contemporar nei, c'erano dubbi, pensò Omar, che tutto ciò di cui aveva bisogno per sapere per i suoi progetti futu ri riposati vicino a casa. "Allora, qual è la tua convinzione?"Chiese Wynnet mentre metteva le tazze e arrostite-vapore bianche e al tavolo. "Mi dispiace, potresti ripeterlo per favore?"Omar ha dichiarc ato. "Certamente.Ho detto, "Allora, qual è la tua		Italian
	Vedendo questo piccolo magazzino di blocchi di informazioni antiche, post-moderni e contemporar nei, c'erano dubbi, pensò Omar, che tutto ciò di cui aveva bisogno per sapere per i suoi progetti futu ri riposati vicino a casa. "Allora, qual è la tua convinzione?"Chiese Wynnet mentre metteva le tazze e arrostite-vapore bianche e al tavolo. "Mi dispiace, potresti ripeterlo per favore?"Omar ha dichiarc ato. "Certamente.Ho detto, "Allora, qual è la tua convinzione"	"convinzione"	Italian

Figure A.4: This table shows some good examples of the LAMBADA tasks for 16-patch-size PIXAR++$_{stage2}$

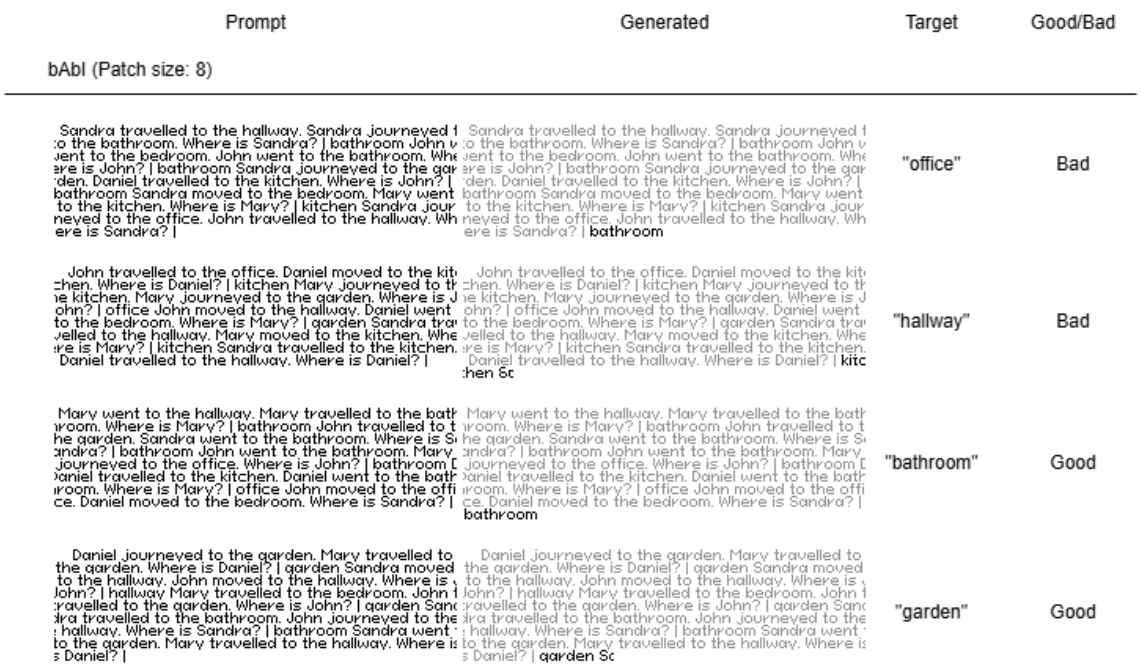
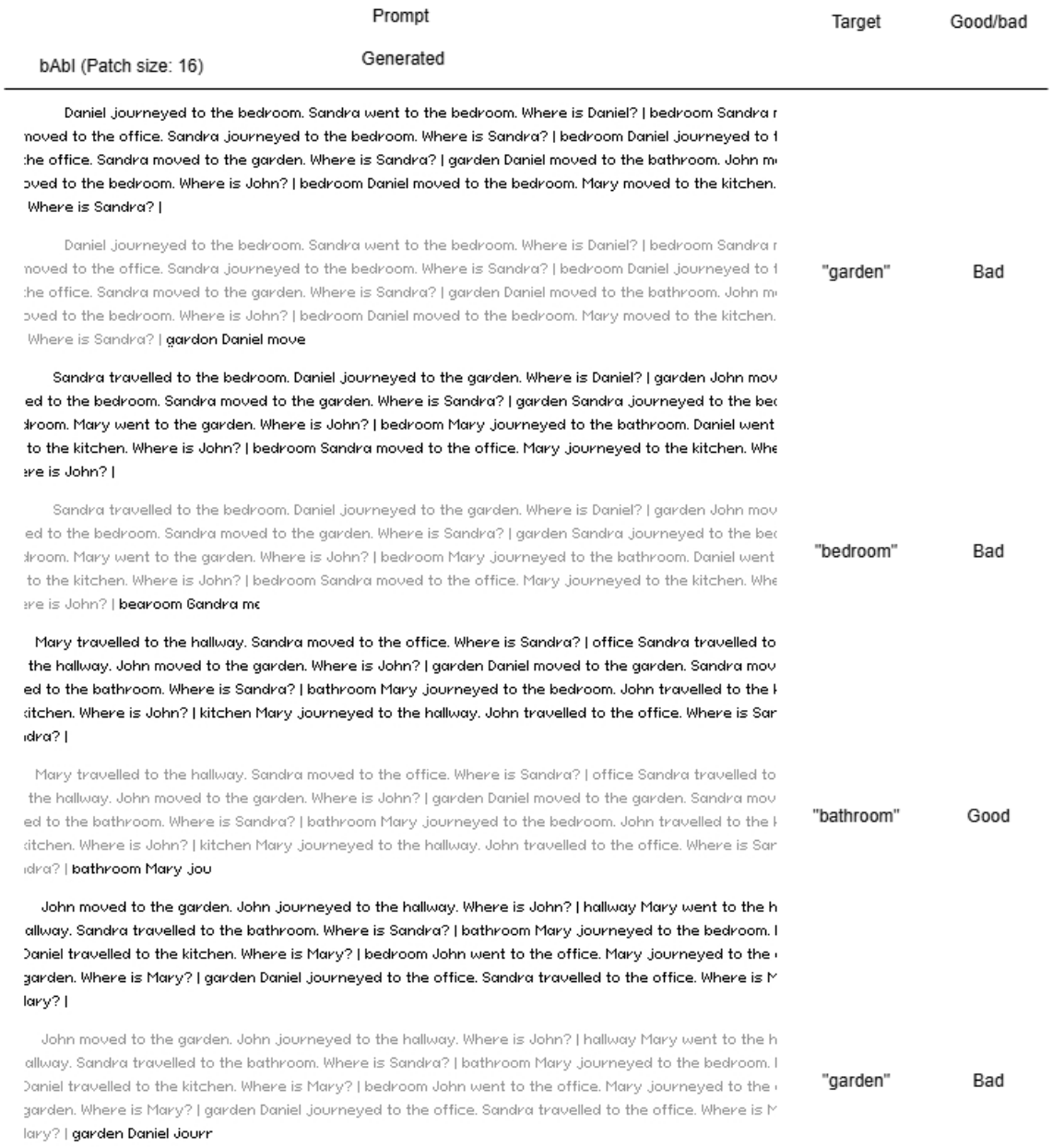


Figure A.5: This table shows some good and bad examples of the bAbI tasks for 8-patch-size PIXAR++$_{stage2}$



bAbI (Patch size: 16)	Prompt Generated	Target	Good/bad
	Daniel journeyed to the bedroom. Sandra went to the bedroom. Where is Daniel? | bedroom Sandra r noved to the office. Sandra journeyed to the bedroom. Where is Sandra? | bedroom Daniel journeyed to t :he office. Sandra moved to the garden. Where is Sandra? | garden Daniel moved to the bathroom. John m oved to the bedroom. Where is John? | bedroom Daniel moved to the bedroom. Mary moved to the kitchen. Where is Sandra? |		
	Daniel journeyed to the bedroom. Sandra went to the bedroom. Where is Daniel? | bedroom Sandra r noved to the office. Sandra journeyed to the bedroom. Where is Sandra? | bedroom Daniel journeyed to t :he office. Sandra moved to the garden. Where is Sandra? | garden Daniel moved to the bathroom. John m oved to the bedroom. Where is John? | bedroom Daniel moved to the bedroom. Mary moved to the kitchen. Where is Sandra? | gardon Daniel move	"garden"	Bad
	Sandra travelled to the bedroom. Daniel journeyed to the garden. Where is Daniel? | garden John mov ed to the bedroom. Sandra moved to the garden. Where is Sandra? | garden Sandra journeyed to the bed droom. Mary went to the garden. Where is John? | bedroom Mary journeyed to the bathroom. Daniel went to the kitchen. Where is John? | bedroom Sandra moved to the office. Mary journeyed to the kitchen. Whe ere is John? |		
	Sandra travelled to the bedroom. Daniel journeyed to the garden. Where is Daniel? | garden John mov ed to the bedroom. Sandra moved to the garden. Where is Sandra? | garden Sandra journeyed to the bed droom. Mary went to the garden. Where is John? | bedroom Mary journeyed to the bathroom. Daniel went to the kitchen. Where is John? | bedroom Sandra moved to the office. Mary journeyed to the kitchen. Whe ere is John? | bearoom Gandra me	"bedroom"	Bad
	Mary travelled to the hallway. Sandra moved to the office. Where is Sandra? | office Sandra travelled to the hallway. John moved to the garden. Where is John? | garden Daniel moved to the garden. Sandra mov ed to the bathroom. Where is Sandra? | bathroom Mary journeyed to the bedroom. John travelled to the k itchen. Where is John? | kitchen Mary journeyed to the hallway. John travelled to the office. Where is Sar idra? |		
	Mary travelled to the hallway. Sandra moved to the office. Where is Sandra? | office Sandra travelled to the hallway. John moved to the garden. Where is John? | garden Daniel moved to the garden. Sandra mov ed to the bathroom. Where is Sandra? | bathroom Mary journeyed to the bedroom. John travelled to the k itchen. Where is John? | kitchen Mary journeyed to the hallway. John travelled to the office. Where is Sar idra? | bathroom Mary jou	"bathroom"	Good
	John moved to the garden. John journeyed to the hallway. Where is John? | hallway Mary went to the h allway. Sandra travelled to the bathroom. Where is Sandra? | bathroom Mary journeyed to the bedroom. I Daniel travelled to the kitchen. Where is Mary? | bedroom John went to the office. Mary journeyed to the garden. Where is Mary? | garden Daniel journeyed to the office. Sandra travelled to the office. Where is M lary? |		
	John moved to the garden. John journeyed to the hallway. Where is John? | hallway Mary went to the h allway. Sandra travelled to the bathroom. Where is Sandra? | bathroom Mary journeyed to the bedroom. I Daniel travelled to the kitchen. Where is Mary? | bedroom John went to the office. Mary journeyed to the garden. Where is Mary? | garden Daniel journeyed to the office. Sandra travelled to the office. Where is M lary? | garden Daniel jourr	"garden"	Bad

Figure A.6: This table shows some good and bad examples of the bAbI tasks for 16-patch-size PIXAR++$_{stage2}$