KU LEUVEN

FACULTY OF SCIENCE

Improving Text Rendering In Image Generation

Medha HEGDE

Supervisor: Prof. Dr. Tinne Tuytelaars
KU Leuven

Co-supervisor: Dr. Jan van Looy
ML6

Thesis presented in
fulfillment of the requirements
for the degree of Master of Science
in Statistics & Data Science

Academic year 2022-2023

Preface

This Master's Thesis is submitted for the completion of the Master of Statistics and Data Science program. Artificial Intelligence is currently witnessing rapid advancements with applications in various fields, Computer Vision being one among them. With this research, we aim to contribute to the existing body of knowledge on image-generation models and their capabilities. I would like to first thank my thesis supervisor, Prof. Dr. Tinne Tuytelaars for her invaluable guidance, expertise and motivation during the entire thesis process. I would also like to express gratitude to the co-supervisor of this thesis, Dr. Jan van Looy, for the formulation of the thesis topic and also his expert opinion and suggestions throughout. Special thanks to Bert Christiaens for taking the time to provide his irreplaceable advice and recommendations in all aspects of this thesis. A token of appreciation to ML6 for allowing me to conduct my analysis with their guidance and supervision. Finally, I thank my family and friends for their unwavering support and encouragement which was the driving force for the completion of this thesis.

Medha Hegde

Summary

The aim of this thesis is to understand and improve the text rendering capabilities of image generation models. Text-to-image generation models are able to create images from text captions that describe them. Current text-to-image generation models such as DALL-E 2 (Ramesh et al., 2022), Stable Diffusion (Rombach et al., 2022) and Imagen (Saharia et al., 2022) use artificial intelligence to create highly realistic images from text captions that describe them. A common issue with these image generation models is that they are unable to correctly render text in an image. The incorrect text generated in images are often misspelled or even illegible. This thesis explores this phenomenon by running a series of experiments that reveal the role of different aspects of the model architecture in the rendering of text in an image. We find that the text encoder used to condition the image generation model is one of the sources of incorrect text rendering and that replacing it with a character-level language model results in improvements.

Chapter 1 introduces the topic of the thesis, defines the problem and lists the research questions we aim to answer. Chapter 2 and Chapter 3 delineate the required background literature necessary for our analysis. We separate our experiments into three parts: Preliminary Experiments, Character Probe Experiment and Main Experiments. The methodology for all experiments is explained in Chapter 4. The purpose of the Preliminary Experiments and the Character Probe Experiment is to explore the nature of the "reading" capabilities of the CLIP text model, while that of the Main Experiments is to expose the most important architectural aspects that have an impact on text rendering such as the text encoder type, text encoder size, model training time, and combined embeddings. We create our own synthetic dataset of text in images for the Preliminary Experiments as well as the Main Experiments.

Chapter 5 presents and discusses the results of these experiments. The Preliminary Experiments suggest that while the CLIP text model has OCR capabilities, it is unable to recognise character positions in a word. The Character Probe model shows an unexpected feature of the CLIP model by demonstrating that it's token embeddings do contain character-level information. The Main Experiments demonstrate that replacing the CLIP text encoder with the character-level large language model ByT5 results in a significant improvement in the text rendering capabilities of the image model. We also find that larger text models, increased training time and combined character-model text embeddings result in improvements. Finally, Chapter 6 concludes the analysis while also identifying limitations, risks and the potential for future work in the context of this thesis.

Contents

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
GAN	Generative Adversarial Networks
NLP	Natural Language Processing
BERT	Bidirectional Encoder Representations from Transformers
BPE	Byte Pair Encoding
GPT	Generative Pre-trained Transformer
T5	Text-To-Text Transfer Transformer
C4	Colossal Clean Crawled Corpus
UTF-8	Unicode Transformation Format - 8 bits
ViT	Vision Transformer
CLIP	Contrastive LanguageImage Pre-training
OCR	Optical Character Recognition
VAE	Variational Autoencoder
GPU	Graphics Processing Unit
RAM	Random-Access Memory
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error

Symbols

$\mathbb{R}^{a\times b}$	Set of Real Numbers of dimension $a \times b$
$\in$	Belongs to
$\sim$	has the distribution (of)
$\coloneqq$	is defined by
$\prod$	The product of
$\mathcal{N}(x; y, z)$	Normal Distribution of variable x, with mean y and standard deviation z
$\sum$	The sum of
$\prod$	The product of

Chapter 1

Introduction

Image generation is the process of using machine learning methods to create new images that are often indistinguishable from human-created images. It applies the field of artificial intelligence on images to learn the patterns underlying the data distribution of these images. Image generation can be unconditional, meaning that the model is trained to generate arbitrary images from the learned data distribution, or conditional, meaning that the model is trained to generate images based on specific conditions or input. Conditional image generation involves providing additional information or constraints to guide the image generation process.

These models, when given additional textual information, are called text-to-image models. Image generation in these types of models has seen tremendous progress over the recent years. A natural language text caption is given as an input and an image matching this caption is generated as an output. The general structure of these models involves two elements - a text encoder to transform the natural language text input to a latent representation and an image generator to produce an image that best reflects

(a) DALL-E 2: "An astronaut riding a horse in a photorealistic style" (OpenAI, 2022)

(b) Imagen: "A brain riding a rocketship heading towards the moon" (Saharia et al., 2022)

(c) Stable Diffusion: "a Van Gogh style painting of a bouquet of pink flowers" (Rombach et al., 2022)

Figure 1.1: Images generated using Text-to-Image models, with their corresponding text captions under each image

the semantic information from the text input.

For the first element - text encoding - Transformer text models (Radford et al., 2021, 2018; Raffel et al., 2020) are most often used because of their state-of-the-art performance on natural language tasks. For the second element - image generation - conditional Generative Adversarial Networks (GANs, (Goodfellow et al., 2014)) were the state-of-the-art, but are being increasingly replaced by diffusion models (Dhariwal and Nichol, 2021) in recent years.

Since 2022, there has been a rapid advancement in the quality and fidelity of the images generated by models such as DALL-E 2 (Ramesh et al., 2022), Imagen (Saharia et al., 2022) and Stable Diffusion (Rombach et al., 2022). The images generated by these models are often very realistic and creative, making it difficult to distinguish them from human-made content, as seen in Figure 1.1.

1.1 Problem Definition

What is text rendering? In the context of image generation, we refer to *text rendering* as the words visible in an image as written or printed text. These can include signboards, posters, documents, tweets or any other image which contains readable-text. Text-to-image models are able to "read" text from the information provided in the text caption and are able to "write" text by rendering the required text in the image, within the context provided by the caption.

(a) DALL-E 2 (Ramesh et al., 2022)

(b) DeepFloyd IF (DeepFloyd, 2023)

(c) Stable Diffusion (Rombach et al., 2022)

Figure 1.2: Examples of Text Rendering Issues: Images generated using the text prompt *"New York Skyline with "Hello World" written with fireworks on the sky."* (caption from Saharia et al. (2022)). The text rendered in the images are incorrectly spelt with repeated characters in (a), jumbled characters in (b) and out-of-word characters ("I" and "M") in (c).

While the images generated by these models are generally of high quality, diverse and similar to the inputted text captions, they often struggle with text rendering in the image. When these models are prompted to generate an image with written or printed text in the image, the text rendered in the image is often incoherent with wrong spelling or just plain gibberish being outputted. Ramesh et al. (2022) mention that DALL-E

2 "struggles at producing coherent text" and one of the limitations of Stable Diffusion as listed here is that the model "cannot render legible text". Sample images generated with incorrect text rendering can be seen in Figure 1.2.

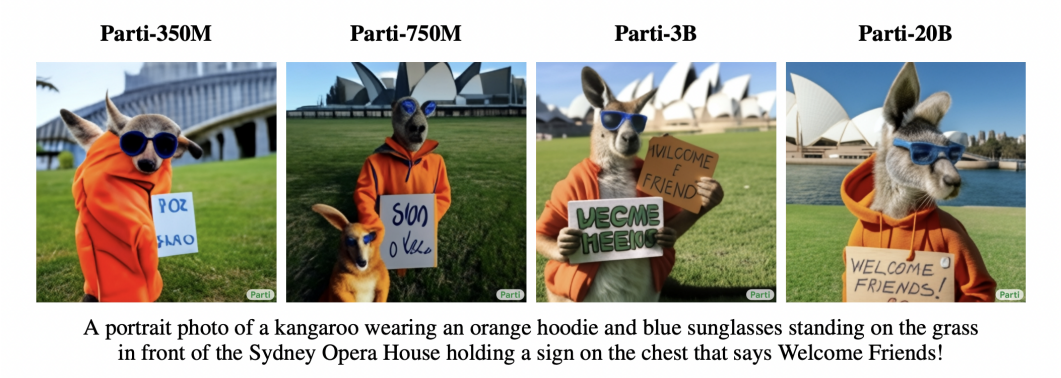


Figure 1.3: Improvement of text rendering with model size, image from Yu et al. (2022)

Figure 1.3 displays images generated using the same text caption by another text-to-image model, the Parti model (Yu et al., 2022), with increasing model size from left to right. We see that the text *"Welcome Friends!"* is not correctly rendered in the first three images. This issue disappears in the right-most image where the model is scaled up to have a much larger number of parameters (20 Billion) than the left-most model (350 million).

Since this issue is seemingly "fixed" simply by scaling up the number of parameters, we can hypothesise that text rendering is a higher-order task for such models. It first learns to generate objects in the image that generally match the text caption, but as the parameters are increased the model is able to generate images that are more fine-grained, specific and accurate to the caption inputted. At lower parameters, the model is able to render something that resembles text but is not actually sensible text, similar to a human writing in a script they have seen, but do not actually know to write: the general shape of characters will be known but accurate spelling and order of characters and words will not. We can say that the model (with a smaller number of parameters) has not yet learned how to write but has only learned the general shape of text in images. As such, our task will be to teach the model to be able to render text in images without increasing the number of parameters in the model.

Therefore, with this Master's Thesis, we will investigate other methods to improve text rendering in image generation models. We propose to do this by first exploring the current nature of image generation models with respect to text rendering, and then by changing aspects of the model architecture to observe its effects on the text rendered in the image.

1.2 Research Questions

Text-to-image generative models have their uses in the fields of art, visual design and other multi-media areas (Yu et al., 2022). It allows lay-people to create high quality art without having to learn and practice the skills necessary to create said art. Since the applications of these models are wide and ever-growing, it is important to investigate and improve them. One of the limitations of these models, as identified above, is incorrect text rendering in images. We propose to answer the following research questions:

1. What are the causes for incorrect text rendering in current image generation models?
2. What are the ways in which text rendering in current image generation models can be improved?
3. Can an improved model be trained using one of the methods identified in Research Question 2?

In this chapter, we introduced our topic of analysis: *"Improving Text rendering in Image Generation"*. Text-to-image models generate images on the basis of textual input, but they often struggle with text rendering, producing incoherent or gibberish text. The goal of this Master's Thesis is to explore the nature of text rendering in image generation models, modify the model architecture, and observe the effects on text rendering in the generated images. By improving text rendering, the overall quality and usefulness of text-to-image generative models can be enhanced.

Chapter 2

Preliminary Background

Previously, we defined our goal with this Thesis and enumerated our Research Questions. This next chapter provides the necessary background knowledge that forms the foundation of our analysis. We introduce the topics of transformers and image generation models, and expand upon the sub-topics most relevant for this Thesis.

2.1 Deep Learning

Machine learning is the process by which computers learn to recognise patterns in data and make predictions from these patterns. It is a multi-disciplinary field, combining the fields of artificial intelligence and statistics. Machine learning models can range from being as simple as a logistic regression model to as complex as a deep neural network. The general goal of such models is to learn from input data (training data) and then to be able to perform well on new, unseen data (test data). This learning process can be supervised (labelled input data), unsupervised (unlabelled input data) or semi-supervised (partially labelled data). Areas where machine learning is used commonly include speech recognition, computer vision and natural language processing.

Neural networks are a subset of machine learning methods that involve neurons or nodes interconnected through multiple layers. They contain an input layer, hidden layers and an output layer, each consisting of multiple nodes. The input data is "fed" through the entire network consisting of nodes that transform it using linear and non-linear activations. This is also known as a feedforward neural network which approximates a function $y = f(x; \theta)$ with input x and parameters θ (Goodfellow et al., 2016). When the number of hidden layers is more than one, the network is referred to as a "deep neural network" and the corresponding learning process is termed as "deep learning". These deep learning algorithms generally require large amounts of input data that can be of any type such as text, images or sound.

2.2 Transformers

Vaswani et al. (2017) originally introduced the Transformer, which is a deep learning architecture that utilises the *attention mechanism* for sequence to sequence tasks such

as translation, summarizing and text-generation, among many others. For example, a transformer translation model takes as input a sentence in English and then outputs the corresponding translated sentence in Dutch. This model architecture was originally applied to natural language processing (NLP) tasks and led to the development of BERT (Devlin et al., 2018) and the GPT models (Radford et al., 2018). Subsequently, this architecture has also been used for computer vision tasks where instead of words, sequences of patches in an image are used for tasks such as image classification, image generation and semantic segmentation.

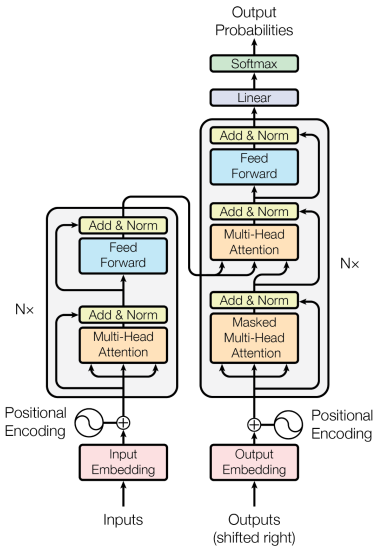


Figure 2.1: Transformer Architecture (Vaswani et al., 2017)

Attention Earlier models such as recurrent neural networks (RNNs) and long short-term memory (LSTM) neural networks used fixed-length context vectors which proved to be a disadvantage for long sentences. Bahdanau et al. (2014) tackled this problem by introducing the concept of attention in RNNs. Their proposed attention mechanism works by determining which parts of the input sequence are relevant for the prediction of the next word. Vaswani et al. (2017) further refined this attention mechanism and defined it using three main matrices: queries Q, keys K and values V. A scaled dot-product attention is used to produce a weighted-sum of values V with weights being the dot-product of the queries Q and keys K, scaled by the sequence length d_k:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Vaswani et al. (2017) employ multi-head attention to parallelly run this attention function by splitting the Q, K and V matrices into different representations of the input sequence and performing the attention mechanism on each split. The outputs of each are then concatenated and projected into the final output space. This allows the model to jointly attend to multiple parts of the input sequence.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$. The parameters are learned during training. Vaswani et al. (2017) use $h = 8$ attention heads with $d_k = d_v = d_\text{model}/h = 64$ for their model.

Encoder-Decoder Transformer models generally work with an encoder and a decoder, both consisting of multiple layers. The encoder extracts features from the input while the decoder produces an output from these extracted features. In the case of an English to Dutch translation model, an English sentence is inputted to the encoder which converts this text into a numerical representation by passing it through multiple encoder blocks. Using the attention mechanism as described above, different weights are associated with different words in the input sequence for the encoder during this process. Vaswani et al. (2017) use 6 layers in their encoder, each consisting of two sub-layers: a multi-head self attention layer and a fully connected feed-forward layer, as seen on the left half of Figure 2.1. Each sub-layer is followed by a residual connection layer and a normalisation layer, to alleviate vanishing gradients and improve training stability.

Similarly, the decoder also has 6 layers, with an additional third sub-layer for each layer which incorporates attention from the output of the encoder. The multi-head attention in the decoder is masked to prevent attending to future positions in the target sequence. Therefore, there are three attention mechanisms at play for the prediction of the next word: encoder self-attention, decoder self-attention and encoder-decoder attention.

Positional information is integrated into the model by using a positional encoding vector that represents the relative positions of each element in the sequence. This allows the model to take into account the position and order of words in a sentence.

2.3 Transformers in NLP

Transformers are now the standard for encoder, decoder and encoder-decoder architectures in the field of natural language processing (NLP). Encoder-only models (Devlin et al., 2018; Radford et al., 2021) are used for representation learning to create embeddings that are useful for classification tasks. Decoder-only models (Radford et al., 2018), also known as auto-regressive models, are particularly well-suited for text generation tasks. The general application of encoder-decoder models (Raffel et al., 2020) in text-to-text tasks involve mapping a sequence of input words to a sequence of target words. These tasks include language translation, text summarizing, text generation, question-answering and reading comprehension, among many others. Models trained for these tasks are also known as language models. Generally, these transformer models have millions of parameters and are referred to as "large language models".

2.3.1 Training

Language models are trained on large amounts of text data using self-supervision to model the patterns and structure of the language it is trained on. The model essentially learns the probability distribution over sequences of words. The data is unlabelled and therefore the model learns from the input data itself. This is known as the "pre-training" of language models which generally use the pre-training objective of masked language modelling (fill-in-the blank) or causal language modelling (next-word prediction). The model is then fine-tuned using labelled data to be able to perform specific tasks such as sentiment analysis, question-answering, part-of-speech tagging and translation.

2.3.2 Tokenization

In order for language models to work with raw text data, it has to be preprocessed before being inputted into the model. The first step of this process is tokenization where the inputted sentence or text document is broken down into individual words, sub-words or characters, known as tokens. Sub-word tokenization schemes solve the issue of large vocabulary size and large number of out-of-vocabulary tokens present in word tokenization schemes. They also do not have the disadvantage faced by character-based tokenization of long sequences of tokens. Frequent words remain whole, while rare words can be split into meaningful sub-words. While there are many tokenization algorithms or schemes, we only discuss those of relevance to this analysis.

Byte Pair Encoding (BPE) This is a subword-tokenization scheme (Sennrich et al., 2015) where the words are split into smaller sub-words based on their frequency in the text corpus. The token vocabulary starts with all the individual characters present in the corpus. Additions are made to the vocabulary by merging the most frequent pairs of existing tokens. This is repeated until a predefined vocabulary size is reached. The sub-words in the final vocabulary set can be combined to form any word from the training corpus. Any new characters are tokenized with a special token, for e.g., $< unk >$. To avoid this, byte-level BPE is used where bytes are used instead as the base vocabulary. This way, all words can be tokenized without the need for the $< unk >$ token.

WordPiece Schuster and Nakajima (2012) introduced the WordPiece tokenization scheme which works in a similar way to BPE except for the mechanism used to choose the next pair to be added to the vocabulary. Instead of choosing the most frequent pair, it divides the pair's frequency by the product of each part's frequency. This results in the selection of a pair which maximises the likelihood of the training data upon merging.

SentencePiece The previous two schemes pre-tokenize by splitting words using delimiters such as white-space. SentencePiece (Kudo and Richardson, 2018) is a pre-tokenization method that instead includes white-space as one of the characters in the base vocabulary, thus treating a sentence as a stream of characters. The remaining tokenization process can remain the same as in BPE, but most models use another process called unigram tokenization (Kudo, 2018). Here, the initial base vocabulary is a large

number of words that is then filtered down to a smaller size. This is done by minimizing the rise in loss (often log-likelihood) due to the removal of a symbol from the vocabulary until the desired vocabulary size is reached.

Character-Based Instead of sub-words, only the base individual characters are retained in the finally vocabulary set. These characters can be represented as is, perhaps as Unicode, or instead represented by bytes. This results in a very small vocabulary size, an advantage which is a trade-off for a large sequence length. Kalchbrenner et al. (2016) and Radford et al. (2017) demonstrate state-of-the-art results using character-based tokenization models.

2.3.3 Models

There are many different transformer language models that can differ in their architecture, tokenization scheme and pre-training objective. Here we discuss the ones most relevant for our analysis.

1. **GPT**

   Radford et al. (2018) from OpenAI introduced the Generative Pre-trained Transformer (GPT) model which was a decoder-only large language model trained on a 4.5GB large text corpus BookCorpus (Zhu et al., 2015), with 117 million parameters. This was the first of many models from the GPT family, with each subsequent model being of larger size. GPT-2 (Radford et al., 2019) was the next iteration, a 1.5 billion parameter model trained on a larger 40GB corpus of 8 million web pages. The GPT-J (Wang and Komatsuzaki, 2021) model is an open-source implementation with 6 billion parameters trained on a 800GB dataset known as The Pile (Gao et al., 2020). These models use only the decoder part of the transformer, often termed as auto-regressive models. They are trained with the objective of predicting the likelihood of the next token (given the previous tokens), also known as causal language modelling. As such, they are generally used for the purpose of text generation, producing coherent and semantically meaningful sentences. They also make use of the BPE tokenization scheme with a vocabulary size of around 50k tokens.

2. **BERT**

   The Bidirectional Encoder Representations from Transformers (BERT) base model (Devlin et al., 2018) is a 110 million parameter encoder-only Transformer model trained on the datasets BookCorpus (Zhu et al., 2015) and Wikipedia. It was trained with the dual objective of masked language modelling and next-sentence prediction. An important distinction from attention mechanism of the GPT models is that BERT uses bi-directional self-attention as opposed to the attention being based only on previous tokens. This model uses only the encoder part of the Transformer architecture (Vaswani et al., 2017), as a result of which it produces a latent representation of the inputted text. These latent embeddings are then used for further down-the-line tasks such as text classification or question-answering.

The model uses the WordPiece tokenization scheme with a vocabulary size of 30k tokens.

3. **T5**

   Raffel et al. (2020) introduced the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer", which is an encoder-decoder transformer language model that models all natural language tasks with a text-to-text scheme. Both the input and output of the model is text. In contrast with the GPT and BERT models, which are decoder-only and encoder-only models respectively, the T5 model uses the both the encoder and decoder parts of the Transformer (Vaswani et al., 2017). It is first pre-trained with an unsupervised training objective similar to BERT, where the model has to predict the masked token. It is then fine-tuned with a supervised objective to perform down-stream tasks such as summarizing, translation and classification. The base model architecture follows that of BERT, resulting in a total of 220 million parameters. There are five size variants of the T5 model, each with an increasing number of parameters, with the smallest model size being 60 million parameters and the largest being 11 billion parameters. Raffel et al. (2020) also created the training dataset, named the C4 (**C**olossal **C**lean **C**rawled **C**orpus) corpus, which is based on the Common Crawl[1] dataset, a large collection of text extracted from web pages. The resulting dataset is a 750GB corpus of English language text spanning all domains enabling the model to generalise on different tasks well. However, due to the extreme size of the dataset, the model trains only on a part of it, resulting in no repeated examples during pre-training. SentencePiece (Kudo and Richardson, 2018) is used to tokenize text with a vocabulary size of 32k tokens.

4. **ByT5**

   The ByT5 model (Xue et al., 2022) is a tokenizer-free version of the T5 (Raffel et al., 2020) model. It follows the same architecture and is also available in five different size variants ranging from 300 million to 13 billion parameters. The only difference from the T5 model is the way the text is preprocessed before being inputted into the model. The ByT5 model does not use a fixed vocabulary set of subwords, but instead utilizes UTF-8 bytes. UTF-8 is a Unicode encoding that is able to encode every character into one to four byte units. The ByT5 model therefore employs character-level encoding to encode the raw text as input, without any preprocessing. It is pre-trained with the same mask prediction objective as T5, modified to work on characters instead of sub-word tokens and the model predicts each character instead of each sub-word token. Another difference from the T5 model is that the ByT5 model is trained on a multi-lingual dataset called mC4 (Xue et al., 2020), a variant of the English-only C4 dataset (Raffel et al., 2020). It contains natural language text from 101 languages scraped from web pages. This token-free method comes at the cost of additional compute, but its advantages include being able to process text from any language and by-pass "complex and error-prone" (Xue et al., 2022) text preprocessing steps.

[1] http://commoncrawl.org

2.4 Transformers in Computer Vision

The Transformer architecture which was originally developed for text, was later also used for images by replacing text-token sequences with image patch sequences in the Vision Transformer (ViT) (Dosovitskiy et al., 2020). The Transformer Encoder is often used alone here to extract features from an image for applications in computer vision tasks such as image classification, object detection, image segmentation and image synthesis. The ViT requires large amounts of image data to be able to provide good results in these tasks.

2.5 Multimodal Transformer: CLIP

CLIP was introduced by Radford et al. (2021), in the paper titled "Learning Transferable Visual Models From Natural Language Supervision". This paper introduces a multi-modal neural network that consists of a vision and text encoder that are jointly trained on a multi-modal dataset consisting of image and text pairs. The goal of the paper is to create a general purpose aligned embedding space over both language and vision representations. By aligning the representations of visual and textual concepts in an unsupervised way, CLIP can be used for a variety of downstream tasks, such as zero-shot classification, optical character recognition (OCR) and action recognition in videos.

2.5.1 Dataset

The motivation for the development of the CLIP model was to leverage natural language supervision to learn visual concepts in images. This allowed for usage of natural language captions of images already abundantly available on the internet. It also allowed easier scaling since new captions do not have to be manually created to obtain more training data. A list of 500,000 query words was used to scrape the Internet by looking for these words in the alt-text of images. This alt-text contains information about the image and usually describes the content of the image. Each query results in an image-text pair, which allowed Radford et al. (2021) to create a large scale dataset without manually annotating the images. The dataset was approximately class-balanced by retrieving up to 20,000 images from each query, resulting in a final dataset of 400 million image-text pairs.

2.5.2 Contrastive Learning

The CLIP model was pre-trained using a simple contrastive objective to predict which pairs actually occurred in a batch of image-text pairs. The contrastive objective involves maximising the similarity of correctly matching image-text pairs and minimising that of all non-matching pairs. The model trains both the text encoder and the image encoder together to learn a joint multi-modal embeddings space. This allows for multi-modal operations between image and text representations and for calculation of a cosine similarity measure. A symmetric cross-entropy loss over this similarity measure is optimized for during training.

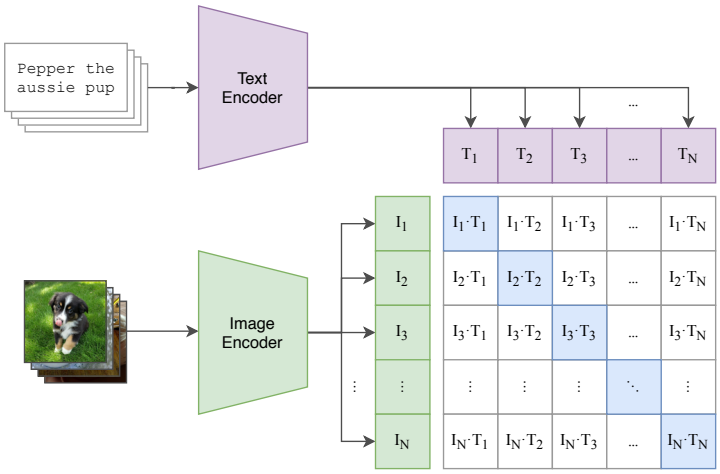


Figure 2.2: CLIP Pre-training Objective, image from Radford et al. (2021)

Figure 2.2 demonstrates this pre-training mechanism where a batch of N image-text pairs is inputted to the model's image and text encoders to generate their embeddings. A dot-product is used to project them into the same multi-modal embeddings space, as represented by the grid of $I_i \cdot T_j (i, j = 1, 2, ...N)$ squares. Diagonal blue squares are the N matching pairs, while the non-diagonal white squares are the $N^2 - N$ non-matching pairs. Cross-entropy loss is used to increase the similarity between the diagonals and decrease that between the non-diagonals.

The CLIP model not only learns the representations between image and text but also within each modality. The text encoder is a language model that generates text embeddings that contain information such that those close to each other in the text embeddings space represent text with similar semantic content. Similarly, the image encoder also generates image embeddings such that those close to each other in the image embedding space represent similar visual content.

2.5.3 CLIP Text Model

The CLIP text model uses the Transformer (Vaswani et al., 2017) encoder architecture, with some modifications. The base text model is a 63 million parameter model with 12 layers and 8 attention heads and an embedding length of 512. The BPE tokenization scheme is used to build the vocabulary set of size 49,612 tokens. A maximum sequence length of 76 tokens was used. Masked self-attention was employed to allow for language modelling as an auxiliary objective and also for the option to use a pre-trained language model instead for training CLIP (Radford et al., 2021). The CLIP text model is thus able to transform raw text into vector representations that, due to the pre-training objective, contain information about visual concepts.

Two different image encoders were tested for the CLIP model: the ResNet (He et al., 2016) and the ViT (Dosovitskiy et al., 2020). The ViT was found to be 3 times more compute efficient than the ResNet model (Radford et al., 2021); this is the CLIP image

model we use in the remainder of this paper.

2.5.4 Zero-Shot Classification

Radford et al. (2021) found that the pre-trained CLIP model demonstrated zero-shot capabilities on multiple downstream tasks such as image classification and OCR. Zero-shot transfer is when a model is able to perform tasks it has not explicitly trained for. In this instance, CLIP is able to perform computer vision tasks it has not explicitly trained for. CLIP is able to do this due to the usage of natural language supervision, the size and diversity of its training dataset and its contrastive pre-training learning objective.

CLIP is trained to represent images and texts in the same embedding space and this is used for zero-shot classification by measuring the cosine similarity between an image and a candidate set of text classes. The classes are represented by natural language texts instead of just the class name alone. The text with the highest similarity score with the image is chosen as the correct class. An important aspect of CLIP's text model is that it requires context for the inputted text caption. For example, it performs better with matching an image of a crane (bird) with the caption "an image of a crane, a type of bird" as opposed to a single-word caption "crane". This could be due to polysemy, where a word can have multiple meanings, and also due to the fact that CLIP rarely sees single-word captions in the pre-training dataset (Radford et al., 2021). It is usually a natural language caption describing the image it is paired with. Similarly, Radford et al. (2021) found that putting quotation marks around text increases performance on OCR datasets.

CLIP's zero-shot performance on OCR datasets can be varied and dependent on the specific type of text in image. Performance on digitally rendered text is measured by classification accuracy on a synthetically-created image dataset of sentences printed in black font colour on a white background. CLIP was found to have higher accuracy on this dataset than on a dataset of hand written text and street view numbers, owing to the lower representation of the latter in the pre-training dataset (Radford et al., 2021). It is also suggested that the model struggles with blurry images and repeated characters. The latter could indicate something about the character-level position information understood by the model.

2.6 Image Generation

Image generation models are deep learning architectures that are able to generate new images by learning the underlying distribution of an image dataset. This is known as generative modelling, where training samples x are drawn from a distribution $p_{data}(x)$, that is unknown. The model learns a distribution $p_{model}(x)$ that is a close approximation of $p_{data}(x)$ to be able to sample generated images that are from the same underlying distribution as the images in the training dataset (Goodfellow et al., 2020).

2.6.1 Variational Autoencoder

Autoencoders (Bank et al., 2020) are neural networks that learn the distribution of the training data using an encoder and a decoder. The encoder learns to transform the input data into a latent representation and the decoder learns to reconstruct the original input from this latent representation. Variational Autoencoders (Kingma and Welling, 2013), based on this architecture, are trained to learn a probabilistic distribution that represents the training data. During sampling, a random point in the latent space is used with the decoder part of the model to generate a new image.

2.6.2 GANs

Realistic image generation using generative modelling was brought to the forefront in 2014 with Generative Adversarial Networks (GANs, (Goodfellow et al., 2014)), which were used to generate very rich, high fidelity images unconditionally. GANs are unsupervised models framed as a supervised learning problem with two sub-models: a generator and a discriminator. The generator outputs synthetic images by capturing the data distribution of the training dataset of images. The discriminator calculates the probability that a given images comes from the real dataset. Both are combined in a zero-sum game fashion where the aim is for the generator to trick the discriminator so that it is unable to distinguish between synthetic and real images. This model was the basis for further advancements throughout the years in areas such as image-to-image translation (Isola et al., 2017), super resolution (Ledig et al., 2017) and style-transfer (Karras et al., 2019).

2.6.3 Diffusion

Diffusion models were shown by Dhariwal and Nichol (2021) to outperform the then state-of-the-art GANs in synthesized image sample quality. They are a class of likelihood-based models that generate images by gradually removing noise from a signal. The model consists of a forward diffusion process, where a Markov chain of Gaussian noise is sequentially added to an image, and a reverse diffusion process, where the model tries to retrieve the original image by predicting the noise added at each step. The paper details improvements to this model architecture in order for it to give the best sample quality. This architecture is the backbone of the DALL-E 2 (Ramesh et al., 2022), Stable Diffusion (Rombach et al., 2022) and Imagen (Saharia et al., 2022) models.

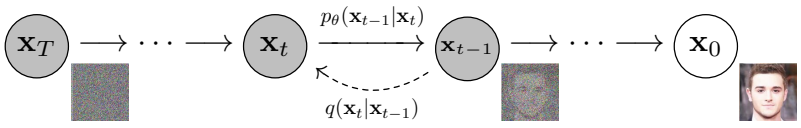


Figure 2.3: Diffusion Model Process, image from Ho et al. (2020)

Forward Diffusion Process Let $q(x_0)$ be the data distribution of the images in a dataset. For any sample $x_0 \sim q(x_0)$, we define a forward diffusion process by adding Gaussian noise for T time steps to produce a sequence of noisy samples $x_1, ..., x_T$. The

step sizes are determined by the variance schedule $\beta_t \in (0,1)$ for $t = 1, ..., T$. The posterior $q(x_{1:T}|x_0)$, called the *forward process*, is the joint distribution of $x_1, ..., x_T$ conditioned on x_0 and is denoted as follows (Ho et al., 2020) :

$$q(x_{1:T}|x_0) \coloneqq \prod_{t=1}^{T} q(x_t|x_{t-1}), \qquad q(x_t|x_{t-1}) \coloneqq \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{2.2}$$

As $T \to \infty$, the distribution of x_T approaches an isotropic Gaussian distribution. An important characteristic of this forward process is that x_t can be sampled at any uniformly distributed time step t, $\forall t \sim \mathcal{U}(\{1, \dots, T\})$ in a closed form (Croitoru et al., 2023):

$$p(x_t|x_0) = \mathcal{N}\left(x_t; \sqrt{\hat{\beta}_t} \cdot x_0, (1-\hat{\beta}_t) \cdot \mathbf{I}\right), \tag{2.3}$$

where $\hat{\beta}_t = \prod_{i=1}^{t} \alpha_i$ and $\alpha_t = 1 - \beta_t$.

Reverse Diffusion Process To generate new samples, a Gaussian sample is used to reverse the Markov chain by predicting the noise added at each time step T. Starting at the standard normal prior $p(x_T) = \mathcal{N}(x_T; 0, I)$, the learned *reverse diffusion process* is the joint distribution $p_\theta(x_{0:T})$ (Ho et al., 2020):

$$p_\theta(x_{0:T}) \coloneqq p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \qquad p_\theta(x_{t-1}|x_t) \coloneqq \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2.4}$$

θ denotes the model parameters, with mean $\mu_\theta(x_t, t)$ and variance $\Sigma_\theta(x_t, t)$ being parameterized by a deep neural network. This network receives the noisy image x_t and the time embedding t as input. $\Sigma_\theta(x_t, t)$ can be set to a constant value (Ho et al., 2020) or can be a learned vector (Nichol and Dhariwal, 2021). The training objective of the neural network is to minimize the negative log-likelihood using the variational lower-bound (Sohl-Dickstein et al., 2015; Croitoru et al., 2023):

$$\begin{aligned} \mathcal{L}_{vlb} = &- \log p_\theta(x_0|x_1) + KL\left(p(x_T|x_0) \| \pi(x_T)\right) \\ &+ \sum_{t>1} KL(p(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)), \end{aligned} \tag{2.5}$$

where KL denotes the Kullback-Leibler (Kullback and Leibler, 1951) divergence between two probability distributions.

As mentioned earlier, Ho et al. (2020) set $\Sigma_\theta(x_t, t)$ to a constant value and re-define the mean $\mu_\theta(x_t, t)$ as a function of noise:

$$\mu_\theta = \frac{1}{\sqrt{\alpha_t}} \cdot \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\hat{\beta}_t}} \cdot z_\theta(x_t, t)\right). \tag{2.6}$$

This allows for a simplified learning objective, as Equation 2.5 is reduced to:

$$\mathcal{L}_{simple} = \mathbb{E}_{t\sim[1,T]}\mathbb{E}_{x_0\sim p(x_0)}\mathbb{E}_{z_t\sim\mathcal{N}(0,\mathbf{I})}\|z_t - z_\theta(x_t, t)\|^2 \tag{2.7}$$

where $\mathbb{E}$ is the expected value, and $z_\theta(x_t, t)$ is the network predicting the noise in x_t (Croitoru et al., 2023).

Types The formulation described above is for a class of diffusion models known as Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020; Nichol and Dhariwal, 2021). There are many other variants of diffusion models that are used for image generation, with the two other main ones being Score-based Generative Models (SGM) (Song and Ermon, 2019) and Stochastic Differential Equations (SDE) (Song et al., 2021).

2.6.4 U-Net

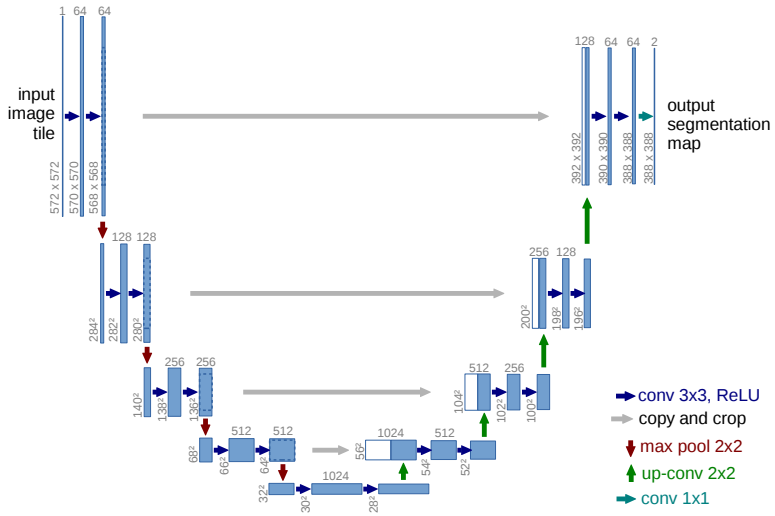


Figure 2.4: U-Net Architecture, image from Ronneberger et al. (2015)

U-Net models (Ronneberger et al., 2015) are often used to model the reverse diffusion process and predict the noise added to an image, also known as denoising. This model is a symmetric convolution neural network consisting of a contracting path that down-samples the input, and an expanding path that up-samples it, resulting in a "U" shape as seen in Figure 2.4. Each down-sampling block consists of: two 3x3 convolution layers, a rectified linear unit (ReLU) and a max-pooling layer for down-sampling. In the expansive path, the image is up-sampled using transpose-convolutions, followed by blocks of two 3x3 convolution layers and a ReLU. The model used skip-connections by concatenating the corresponding features from the contracting path (grey arrows in Figure 2.4). For unconditional image generation, this U-Net architecture is modified to use time step embeddings that contain information about the time-step of the diffusion process. These time embeddings are usually expressed as multiple sinusoidal functions.

These embeddings are linearly incorporated into the U-Net using the self-attention mechanism as in 2.1, making it a time-conditional U-Net.

For conditional image generation, such as text-to-image models, the time-conditional U-Net is conditioned on text embeddings via the cross-attention mechanism (Vaswani et al., 2017). Cross-attention is the same as self-attention but with a different sequence instead of itself, here the image is "attended to" by the text embeddings. These text embeddings are generated using text encoders and then concatenated to the time-step embedding. This model, represented as $z_\theta(x_t, t, y)$, thus learns the conditional distribution of the noise as conditioned on time-step t and text input y.

2.6.5 Text-to-Image Models

DALL-E 2 Ramesh et al. (2022) from OpenAI introduced the DALL-E 2 model that combines a diffusion model and CLIP to create a text-to-image model that can generate an image from an inputted text. It consists of two components: a *prior* that produces CLIP image embeddings conditioned on text captions, and a *decoder* that produced images conditioned on these CLIP image embeddings. The decoder is a 3.5 billion parameter diffusion model based on the Transformer (Vaswani et al., 2017) decoder architecture. This decoder produces 64x64 images, which are then upsampled using two diffusion models to increase the image size to 1024x1024. Ramesh et al. (2022) mention that the model "struggles at producing coherent text" and they hypothesise that this could be due to the fact that CLIP text embeddings do not precisely encode spelling information. Additionally, it is mentioned that the CLIP uses BPE tokenization which obscures the spelling of words.

Stable Diffusion Stability AI's Stable Diffusion model is a latent diffusion (Rombach et al., 2022) model that uses a variational autoencoder (VAE) (subsection 2.6.1) to transform images from the pixel-space to a latent-space before applying the diffusion process. A VAE decoder is used to convert this U-Net-generated latent embedding back to the pixel-space. This allows for the advantage of lower computational requirements for training and generation. It uses only 890 million parameters, compared to the 3.5 billion used by DALL-E 2. The CLIP text encoder is also used to condition on text embeddings using the cross-attention mechanism. Stable Diffusion was trained on a subset of LAION-5B (Schuhmann et al., 2022), a dataset consisting of 5 billion image-text pairs scraped from the internet. This subset consists of images filtered by an "aesthetic score" to include higher quality images. This likely resulted in the exclusion of text-heavy images and hence the model was not trained on a representative set in that regard. Unlike DALL-E 2, Stable Diffusion has made the source code publicly available along with the pre-trained weights.

Imagen Google's Imagen (Saharia et al., 2022) model is also a diffusion model, that uses a frozen T5 model (Raffel et al., 2020) for the text encoder instead of CLIP, which they claim results in a greater increase in fidelity and text-alignment than that from just increasing the scale of the diffusion model. The model uses the U-Net architecture for its base 64x64 model (2 billion parameters) as well as for its two super-resolution

models that upscale the image to 1024x1024 resolution. The text embeddings are used for conditioning using self-attention by adding it to the time-step embeddings as well as via cross-attention. The 4.5 billion parameter T5-XXL model, which is the largest T5 variant, is shown to perform better for text-conditioning than the 123 million parameter CLIP model, based on preferences of human raters. They compare Imagen to DALL-E 2 on multiple categories, including "ability to generate quoted text" and their model is significantly preferred in this specific category. The T5 model is attributed to be the driver of this improvement, demonstrating that the T5 model is better than CLIP about 75% of the time (Saharia et al., 2022). Overall, it appears Imagen suffers less from the issues in text rendering than DALL-E 2 and Stable Diffusion.

e-Diff-I Balaji et al. (2022) from NVIDIA, released their model eDiff-I which consists of an ensemble of diffusion models that utilise a combination of CLIP embeddings and T5 embeddings for text conditioning. Multiple diffusion models are trained for the denoising process at specific noise intervals. This model follows the common structure of a base 64x64 model along with two super-resolution models. Conditioning using the text embeddings is performed in a similar fashion as in Stable Diffusion (Rombach et al., 2022) and Imagen (Saharia et al., 2022). Balaji et al. (2022) show that the inclusion of T5 embeddings removes the issue of incorrect text rendering as seen in DALL-E 2 and Stable Diffusion, both of which use CLIP embeddings only.

This Preliminary Background chapter provided an overview of deep learning models such as transformers and diffusion models used in the realm of image generation. The architecture and the applications of specific models such as CLIP, T5, ByT5, Stable Diffusion and Imagen were also explored. In the next chapter, we will survey the existing research about text rendering in image generation models.

Chapter 3

Related Works

The previous chapter delineated the essential background knowledge with respect to this Thesis. Here, we explore the current works looking into the text rendering capabilities of image generation models.

Goh et al. (2021) investigate how CLIP (Radford et al., 2021) encodes visual and text data by using typographic attacks, and found that the same neurons are fired for similar concepts. For example, the same set of neurons are fired for the colour yellow and for images of a banana. They found a set of "typographic" neurons that are fired when "reading" text in images. They say that despite being able to "read" the text, the model learns an abstract representation of the text, akin to a child writing words it does not know how to spell yet (Goh et al., 2021). Materzyńska et al. (2022) investigate the CLIP representation of words in images to see if its OCR capabilities are separable from the natural images represented by those words. They are able to isolate the spelling capabilities of the model but find that while it does recognise the words in the image, it does not spell perfectly.

Daras and Dimakis (2022) explore the hidden vocabulary of the DALL-E 2 (Ramesh et al., 2022) model. They investigate if the gibberish text rendered in images produced by the model is not random, but is an internal "hidden" vocabulary of the model. They do this by re-inputting the gibberish text into the model to output images that match the original text description. However, this finding is not consistent and does not generalize to all words. Building on this work, Millière (2022) proposes two alternative methods to generate specific imagery from seemingly nonsensical words. Their proposed method, called macaronic prompting, is discovered as a reliable method to combine word chunks from different languages to create a to-human incoherent word but is seemingly coherent to the DALL-E 2 model. This further suggests that the BPE encoding (Section 2.3.2) used by CLIP is the cause of the incoherent text being rendered in images.

Saharia et al. (2022) compare the Imagen model with DALL-E 2 on multiple categories, including "ability to generate quoted text" and their model is significantly preferred by human evaluators in this specific category. The T5 model used for text-conditioning is attributed as the driver of this improvement, demonstrating that the T5 model is better than CLIP about 75% of the time (Saharia et al., 2022, Figure A.7). Overall, it appears Imagen suffers less from the issues in text rendering than both DALL-

E 2 and Stable Diffusion (Saharia et al., 2022) due to the usage of a frozen language model in place of CLIP for text conditioning.

A blog post (nostalgebraist, 2022) published before the release of DALL-E 2, documents an exploration of improving text rendering in image generation models. The model was modified with transcription conditioning by using a transformer encoder with T5 relative position embeddings and character-level tokenization. This method is successful at improving spelling of words in images created by a model with the remaining architecture being similar to the DALL-E 2 model. However, this exploration was done only on images from Tumblr[1].

Similarly, Balaji et al. (2022) use T5 text embeddings for conditioning their image generation model and suggest that their inclusion reduces the issue of incorrect text rendering as seen in DALL-E 2 and Stable Diffusion. However, they still retain the CLIP embeddings too and use it in combination with the T5 embeddings to leverage the image generation benefits of both text encoders.

Liu et al. (2022), in their paper titled *"Character-Aware Models Improve Visual Text Rendering"*, investigate the use of character-aware text encoders for conditioning in image generation to improve text rendering. They compare the T5 and ByT5 text encoders and find that the latter outperforms the former in the task of accurately spelling text in images generated by text-to-image diffusion models. However, this gain in text accuracy comes at the cost of image-text alignment. As such, they propose a combination model that uses both encoders which is demonstrated to maintain alignment while improving the text rendering ability. We use this work as inspiration for our experiments as detailed in Section 4.3.

This chapter explored previous works that investigate the issue of text rendering in image generation models. Specifically, we cited works that looked at the CLIP model's ability to "read", DALL-E 2's "hidden vocabulary" and those looking specifically at improving text rendering. We use these works to be the inspiration for our Methodology as detailed in the next chapter.

[1] https://www.tumblr.com/

Chapter 4

Methodology

The preceding two chapters provided an understanding of the fundamental knowledge needed to build upon as we move forward. Our analysis is designed as a series of experiments that are divided into three sections: the Preliminary Experiments, the Character Probe Experiment and the Main Experiments. In this chapter, we provide the details of the research purpose and the experimental set-up of each experiment.

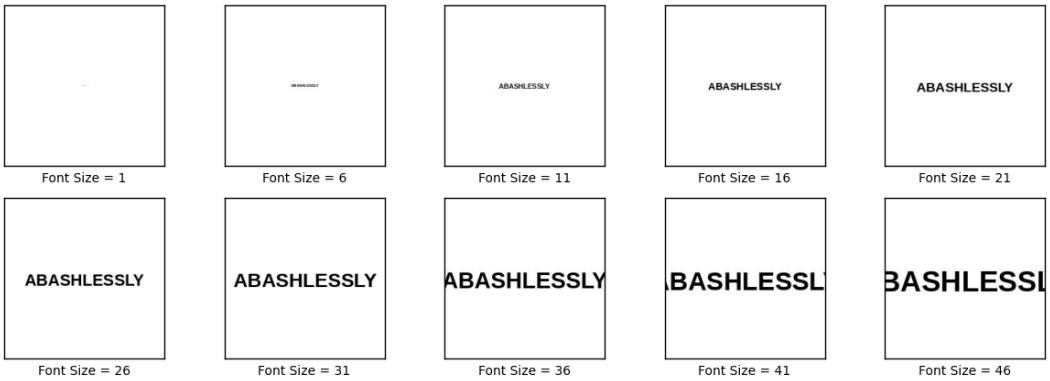
4.1 Preliminary Experiments

4.1.1 Purpose

We formulate four preliminary experiments to demonstrate the nature of CLIP's (Radford et al., 2021) text encoding and how it is affected by changing certain variables such as text size and character position. We would like to observe the "reading" capabilities of the model by looking at the similarity scores between images and text captions.

4.1.2 Method

A set of about 1 million words was extracted from Wiktionary (Ylonen, 2022) and preprocessed by only retaining single words with no punctuation or symbols and with word length less than 30 characters. A random sample of 1,000 words from this set is used for the experiments. The general format of the experiments is the same, a word from the Wiktionary dataset is first printed on an image in black font colour. This image is then used to create CLIP image embeddings and then is compared to the text embedding of the caption 'an image of the word "<WORD>"'. The word is placed in quotation marks in the text caption to increase the cosine similarity scores. These image-text similarity scores are referred to as "CLIP scores". Figure 4.2 demonstrates this process. Both CLIP text and image embeddings are calculated using clip-retrieval (Beaumont, 2022). Words are printed in upper-case letters and in black font colour, on images of size 256x256. Sample images used for these experiments can be seen in Figure 4.1.



(a) Experiment 1: Increasing Font Size, Blank Background

(b) Experiment 2: Increasing Font Size, Image Background

(c) Experiment 3: Text Scrambling, In Image

Figure 4.1: Sample Images used for Preliminary Experiments

Experiment 1: Increasing Font Size, Blank Background The word is printed on a blank image with a white background. Each word is printed 10 times on separate images with increasing font size ranging from 1 to 46, with increments of 5. The goal is to see how well rendered text and text caption are matched with no background noise. This allows for the simplest formulation to test the OCR capabilities of the CLIP model by eliminating any chances of "distraction" from the rendered text in the image. The increasing font size for each word allows us to see how large the rendered text has to be in order for CLIP to be able to "read" the text in the image.

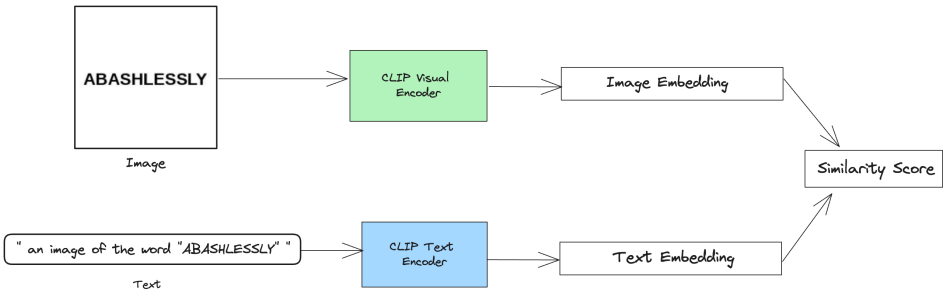


Figure 4.2: Preliminary Experiments Process. Experiment 4 compares two text embeddings to each other instead of an image embedding.

Experiment 2: Increasing Font Size, Image Background The word is printed on a random, unrelated image. Each word is again printed 10 times on separate images with increasing font size ranging from 1 to 46, with increments of 5. Similar to Experiment 1, we use this experiment to test the OCR capability of the CLIP model but we use a background image instead of no background. This is to observe the impact of a background image on the visual information stored in the CLIP image embeddings. It allows us to measure the ability of CLIP to retain information about the rendered text without being "distracted" by the visual content of the background image.

Experiment 3: Text Scrambling, In Image The word is printed on a blank image with a white background, but with increasing amount of scrambling in each image. The percentage of letters being switched ranges from 0% (original word) to 100% (all letters switch positions randomly), with increments of 25%. This experiment is designed to measure how and if CLIP's visual representation of the image encodes character-level information about the rendered text. We can observe the impact of changing the order of characters in the word on the OCR ability of CLIP. If there is no significant difference in the CLIP representation of the original word and that of the scrambled word, it would be an indication that character-level position information is not included in the CLIP embeddings.

Experiment 4: Text Scrambling, Text Only The same scrambling scheme as in Experiment 3 is used, but the text is not printed on an image. We do not use the CLIP image embeddings in this experiment due to the fact that many image-generation models (Rombach et al., 2022; Saharia et al., 2022) only use the CLIP text model during conditioning, without using the CLIP image model. Since only text embeddings are compared to each other in this experiment, we repeat it again on another text encoder: the ByT5 model (Xue et al., 2022). Using this experiment, we again want to observe the amount of character-level information, if any, embedded in the text embeddings of CLIP and ByT5. We use the ByT5 model since it is a character-level model and we expect the text embeddings of scrambled words to be more dissimilar to that of the original word, compared to CLIP.

4.2 Character Probe Experiment

4.2.1 Purpose

The paper *"What do tokens know about their characters and how do they know it?"* by Kaushal and Mahowald (2022) looks into whether pre-trained language models that use subword tokenization schemes encode character-level information. They do this by designing a number of experiments that probe the token embeddings of several pre-trained language models, such as those from the GPT [1] and BERT [2] families. The goal is see how well a character's presence in a word can be predicted by the token embeddings of the language model. By training a linear binary classifier on the model's frozen token embeddings, they concluded that they do in fact contain character-level information. They also suggest that in general, larger models perform better at this task.

We run Experiment 1 from their paper on the CLIP model to observe if the results are in line with their findings. We would also like to investigate the amount of character level information encoded in CLIP's token embeddings. Additionally, we run this experiment on the T5 model (Raffel et al., 2020) since this model is used in the the later Main Experiments [4.3]. We do not include the ByT5 (Xue et al., 2022) model since the experiment is formulated to use the vocabulary set of each model as generated by their respective tokenization schemes. Since ByT5 does not use any tokenization, it cannot be used for this experiment.

4.2.2 Method

We follow the methodology of Kaushal and Mahowald (2022), visualised in Figure 4.3. A Multi-layer Perceptron (MLP) classifier is trained on the text model's frozen token embeddings to predict if a character is contained in a token or not. For e.g., it will predict 1 (yes) or 0 (no) if the letter *a* is in the token *party* and not in the token *cool*. As such, a separate dataset is created for each letter in the English alphabet. The dataset is equally balanced to contain both positive and negative label tokens, i.e, tokens that contain the letter as well as tokens that do not. A train-test split in an 80-20 ratio is then created. A control task is also run with randomly-initialised token embeddings. The model's performance is evaluated using the weighted aggregate F1-score on the test set. We use two variants of the CLIP model - CLIP-*Large* and CLIP-*Base*, which differ only in number of model parameters due to increase in text embedding length.

4.3 Main Experiments

The previous experiments were solely conducted on the text models used for conditioning during the image generation process. We now move to testing the text rendering capabilities of current image generation models. To do this, we formulate a series of experiments that make use of a modified version of the U-Net used by Stable Diffusion[1]. We train this model from scratch on our own dataset to simplify the training task

[1] https://github.com/Animadversio/DiffusionFromScratch

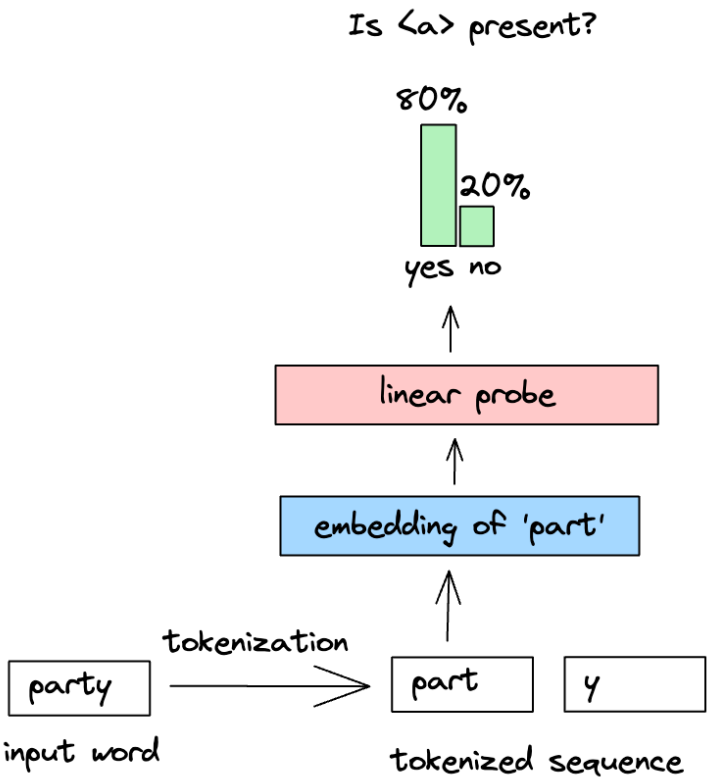


Figure 4.3: Character Probe Process, visualised for a single input word.

and reduce compute required by general purpose models like Stable Diffusion. Through the experiments detailed below [4.3.4], we wish to identify which aspects of the model architecture have an effect on text rendering and the nature of this effect.

4.3.1 Model Architecture

We use the same model architecture as that of the U-Net used by Stable Diffusion. However, we scale the model down to be able to run the entire training with our time and compute constraints. The base model conditional on the CLIP text encoder we use is reduced to 2 million parameters from the original model's 870 million. The modifications made are listed in Table 4.1. Another important difference between our model and the Stable Diffusion model is that we do not make use of the Variational Autoencoder (VAE) to generate latent image embeddings. Thus, we do not make use of the latent diffusion process for our experiments.

4.3.2 Data

Word Buckets We follow the methodology of Liu et al. (2022) to replicate their Wikispell benchmark at a smaller scale. A set of 1 million words extracted from Wiktionary (Ylonen, 2022) was first cleaned by only retaining single words with no punctuation or

Model Element	Stable Diffusion	Ours
No. of colour channels	4	1
No. of Residual Blocks	2	1
No. of Attention Heads	8	4
Base Channel Count	320	16
Text Embedding Length	768	512
Time Embedding Length	1280	256
Total No. of Parameters	**870M**	**2M**

Table 4.1: U-Net Model Architecture Modifications

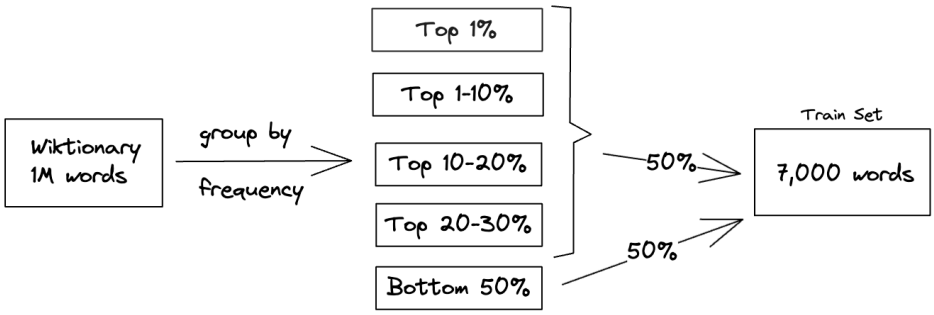


Figure 4.4: Wiktionary Word Buckets creation visualised. An additional non-overlapping test set of 85 words sampled uniformly from each bucket is created.

symbols and with word length less than 10 characters. The words were then grouped into buckets based on their frequency of occurrence in the English sub-corpus of the mC4 corpus (Xue et al., 2020). This sub-corpus contains 700k natural language text documents with a total of 263M words. We create 5 frequency buckets: Top 1%, 1-10%, 10-20%, 20-30%, and Bottom 50%. This means that the Top 1% bucket contains very common words such as "*and*" and "*the*" while the Bottom 50% bucket contains very rare words that the text encoders themselves may not have seen during pre-training. A training set of 7,000 words is created by sampling half from the Bottom 50% bucket, while the other half is sampled proportionally to the size of each remaining bucket. Additionally, a test set of 85 words with 17 words in each bucket is created to be used for evaluation. There is no overlap between the train and test set of words. A visualisation of this word-bucket process can be seen in Figure 4.4.

Image-Text Dataset The words in our training set are then converted to images to create a synthetic dataset of 64x64 images with printed text. This process can be visualised in Figure 4.5. Words are printed in upper-case with white font colour onto a black background. Additionally, a file containing the corresponding text captions for each word is generated to create a dataset of image-text pairs ready for training. We generate only black and white images (1 colour channel) instead of coloured images (3 colour channels for R, G and B). The caption we use for all images follows the format *'a*

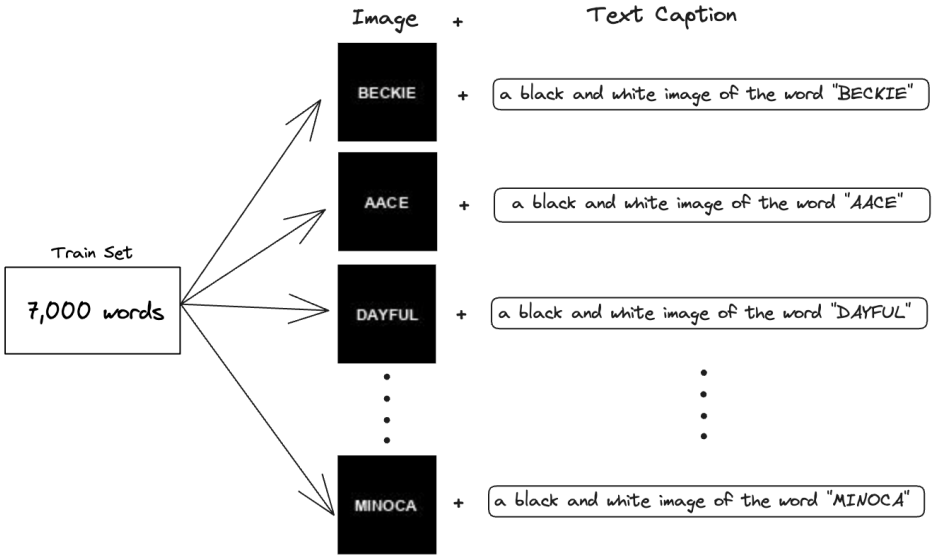


Figure 4.5: Image-Text Dataset Creation

black and white image of the word "<WORD>"', where the word is specified in upper case and within quotation marks. A sample of the training images can be seen in Figure 4.6.

Dataset Size We use a training dataset of 7,000 of image-text pairs. In contrast, the Stable Diffusion model is trained on a dataset of over 2 billion image-text pairs. The reason for this large training dataset size is to be able to learn the largest range of image distributions possible. This enables models like Stable Diffusion to generate images of any class or classes as seen in the large training dataset. On the other hand, since our training dataset consists only of words printed on a black background, with no variation in font colour, font style or font size, the model is able to focus specifically on the task of "writing" letters in a word. This way, we are able to concentrate on this single task of text rendering without the additional task of learning the complexities of natural images. We are thus able to identify the aspects of the model architecture that effect its text rendering capabilities. We found that our small dataset size of 7,000 images is sufficient for our 2 million parameter model to generate images that contain coherent letters and words.

4.3.3 Training

The diffusion process is visualised in Figure 4.7. As detailed in Section 2.6.3, image generative diffusion models are trained to predict the noise added to an image. Our training objective for our U-Net model is also the same. We use $T = 256$ time steps to model the diffusion process. A simplified loss function that resembles Mean Squared Error (MSE) of predicted noise is used to guide the model during training. The maximum number of tokens generated for any text encoders used is set to 77. A batch size of

Figure 4.6: The corresponding text captions are *'a black and white image of the word "<WORD>"'*, with the corresponding word replaced for each image.

128 images is used for all experiments and the models are trained for 150 epochs, both hyperparameters determined by compute and time constraints. We use the Adam (Kingma and Ba, 2014) optimizer with a learning rate of 0.001 set on a learning rate scheduler to linearly drop to 0.0002 for the first 80 epochs, after which it remains constant at that rate.

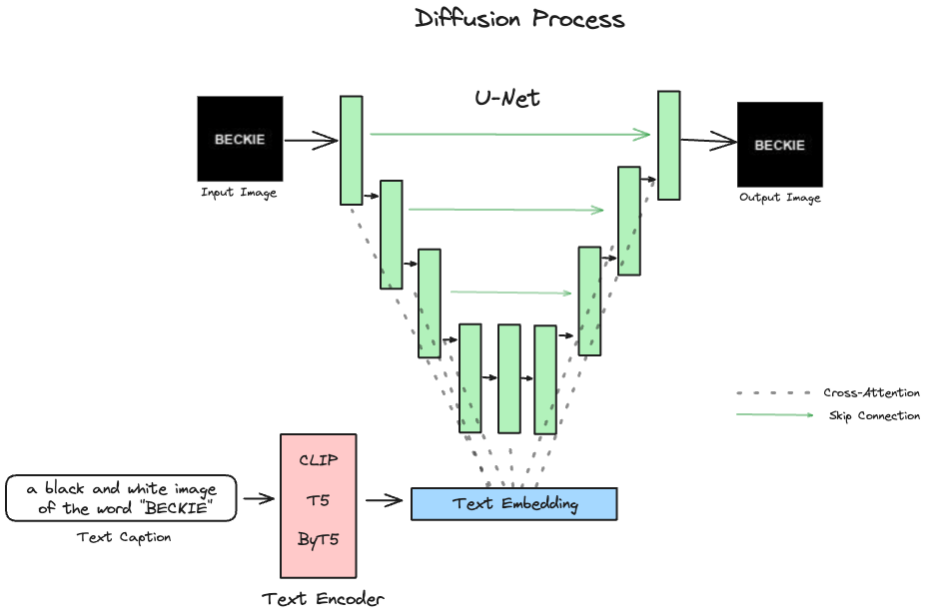


Figure 4.7: Main Experiment Diffusion Process, visualised for one input image

After every training epoch, images are sampled using the sample text caption *'a black and white image of the word "HELLO"'* in order to visualise the model's learning process. This can be seen in Figure 4.8 where we see how the model learns how to "write" after 150 epochs. At the initial epochs, the general placement of white pixels within the same vertical plane is correct, but it does not yet resemble a word. Around epoch 40 we can differentiate between the characters as individual letters are rendered, while it does not yet spell the word "HELLO". As training progresses, the letters rendered move closer to those in the word and eventually at epoch 150 we see the model is able to clearly render the word "HELLO". Images sampled at every epoch during training can be seen in the

Appendix [A.5].

Figure 4.8: Model with CLIP ViT-B/32 Training Process Visualised. Images generated with the prompt *'a black and white image of the word "HELLO"'*. At the start of the training process, the letters aren't very clear, at epoch 40 we can identify individual letters and finally at epoch 150 it is able to clearly render the word "HELLO".

4.3.4 Experiments

Text Encoder Type The first architectural aspect we would like to test is the type of text encoder used for conditioning. As such, the first experiment we run is by switching the CLIP text encoder for different text encoders. We propose two large language models to be used here: the T5 (Raffel et al., 2020) and the ByT5 (Xue et al., 2022) models. As detailed in Section 2, they are transformer based encoder-decoder architectures trained in a text-to-text format for various NLP tasks such as translation, classification and question-answering. We make use of only the text encoder part of both models since we just require embedding representations of our text for conditioning.

The choice of these two models is driven by previous works where they demonstrate that pre-trained large language models can be used for conditioning image generation models (Saharia et al., 2022; Balaji et al., 2022; Liu et al., 2022). Saharia et al. (2022) compared the image generation capabilities of a model conditioned on CLIP with a model conditioned on T5, while Liu et al. (2022) compare that of T5 and ByT5. They have shown that these language models are just as, if not more, effective for text-conditioning in image generation as an image-text multi-modal model like CLIP. Our contribution to this area of research is an experiment that directly compares these three models on the specific task of text rendering. We keep all other model parameters fixed, only changing the source text embeddings used for conditioning during training.

Model Aspect	CLIP	T5	ByT5
Encoder Type	Image and Text	Text	Text
Tokenization	BPE	SentencePiece	None
Pre-training Objective	Image-text Alignment	MLM	MLM
Smallest Variant	63M / 512	60M / 512	300 M / 1472
Larger Variant	123M / 768	220M / 768	582M / 1536

Table 4.2: Comparison of the Text Encoders used in our Main Experiments. The values for the bottom two rows represent the number of parameters / text embedding length. MLM = Masked Language Modelling. The smallest variants of each model is used in the Text Encoder Type Experiment, while the larger variants are used in the Text Encoder Size Experiment.

The CLIP, T5 and ByT5 models differ in multiple ways. The first is, as mentioned above, that CLIP is a multi-modal transformer model consisting of both an image encoder and a text encoder, while the latter two models are language transformer models consisting of a text-only encoder and decoder. With regard to tokenization, the CLIP model makes use of BPE, the T5 model uses SentencePiece with Unigram, while the ByT5 model uses no tokenization and operates directly on characters instead. The pre-training objective of CLIP is image-text alignment, which is why it was originally proposed as an appropriate model for conditioning text-to-image generation models. On the other hand, T5 was pre-trained with a generalised Masked Language Modelling objective, while ByT5 was pre-trained on a modified version of this. All three models are available in multiple size variants, but for this first experiment, we make use of the smallest variant of all three text encoders. As such, we use the CLIP ViT-B/32[2], T5-Small[3] and ByT5-Small models[4]. The number of parameters is 63 million, 60 million and 300 million, respectively, and their embedding lengths are 512, 512 and 1,472.

Text Encoder Size The next aspect of the model architecture we wish to test is the size of the text encoder as defined by the number of trained parameters. We want to observe the effect of increasing just the text encoder size on the quality of text rendered by the diffusion model. This is an important architectural aspect to investigate because we are using frozen text encoders for conditioning, which means that we are not training these model parameters. This implies that for image generation model training, the text encoder can be easily scaled up for larger variants without requiring an proportionally large increase in compute. This is one of the main findings of Saharia et al. (2022) where they also concluded that the quality of images generated improves with scaling of the text encoder.

With our experiment, we investigate if this finding holds true for all three of our text encoder models: CLIP, T5 and ByT5. The CLIP text model is available in two sizes, CLIP ViT-B/32 (*Base*) and CLIP ViT-L/14 (*Large*), so we opt for the latter.

[2]https://huggingface.co/openai/clip-vit-base-patch32
[3]https://huggingface.co/t5-small
[4]https://huggingface.co/google/byt5-small

The embeddings length is increased to 768 from 512 and the number of parameters in the model is doubled to 123 million. The T5 and ByT5 models have 5 sizes - *Small, Base, Large, XL and XXL* - ranging from 60 million to 11 billion parameters for the T5 model and from 300 million to 13 billion parameters for the ByT5 model. Performance across multiple benchmarks for both models increases with increase in model size. For our experiment, however, we use the *base* versions of both due to compute and time constraints. The T5-Base model has 220 million parameters with text embedding length 768, while the ByT5-Base model has 582 million parameters with embedding length 1,536.

The T5 and ByT5 models are scaled up by increasing the number of layers in their models as well as by increasing the length of the text embeddings outputted; the models are scaled by width as well as by depth. On the other hand, CLIP is only scaled up by width and not by depth since the model's performance on the task of image-text alignment was found to be less influenced by the size of the text encoder (Radford et al., 2021).

Training Time Kaplan et al. (2020) say that smaller models need more steps to train than larger models. Among the models tested in so far, the CLIP ViT-B/32 is the smallest. In this experiment, we continue training this model for an additional 650 epochs (after the initial 150 epochs) to observe the impact on performance. We would like to observe whether the model's performance increases with increased training steps and if this performance is comparable to that of the other models from our previous experiments that use large language models as text encoders. The model is evaluated periodically during this extended training to observe changes at multiple checkpoints between 0 and 800 total epochs.

Combined Embeddings Lastly, we run the text generation task using combined text embeddings of the smaller variants of our three text encoders - CLIP, T5 and ByT5. Balaji et al. (2022) were the first to use a combination of the T5 and CLIP text embeddings to condition an image generation model. They suggest that CLIP aids in the overall appearance of images while T5 is able to better guide the model on the fine-grained details. They use the CLIP ViT-L/14 and the T5-XXL models' embeddings and use their concatenation during conditioning via cross-attention. They also found that the combined embeddings perform better than either text encoder alone.

Similarly, Liu et al. (2022) use a concatenation of T5-XXL and ByT5-Small text embeddings to condition their image generation model. They use the smallest variant of the ByT5 model in order to result in a small increase in model size while also making the model "character-aware". Their results demonstrate that this combined model is able to match the text rendering capabilities of models conditioned only on larger ByT5 models (XL and XXL). However, this combined model performs better than the ByT5-only models when evaluated on image-text alignment.

We follow the same methodologies of Balaji et al. (2022) and Liu et al. (2022) of concatenating embeddings for conditioning to observe the impact on text rendering capabilities of our models. This results in three models with the following combinations

of embeddings: CLIP-T5, CLIP-ByT5 and T5-ByT5. We do this to demonstrate the possibilities of leveraging the image generation advantages of each individual model.

4.3.5 Evaluation

We use Optical Character Recognition (OCR) to evaluate our models, seen in Figure 4.9. Using the test set of words from Section 4.3.2, we generate 100 images for each word. EasyOCR[5] is used to detect the word in the generated images. The total number of words (expressed by percentage) correctly spelled by the model is used as an evaluation metric to compare model performance in this task of text rendering. It should be noted that EasyOCR sometimes does not accurately read words from the image but since the same inaccuracy will affect all models equally, we are able to use it to compare performance. Since this OCR score is a percentage count of correctly spelled words, its underlying distribution is a binomial distribution where each image generated can be considered a Bernoulli trial with success when the word is spelled correctly. Since for each word, we generate 100 images, we can model the final score (a proportion) as a binomial distribution. As such, we will use the standard deviation of a binomial distribution to calculate the confidence intervals for the mean proportion of correctly spelled words. We use the normal approximation of the binomial distribution formula (Morisette and Khorram, 1998) to do this.

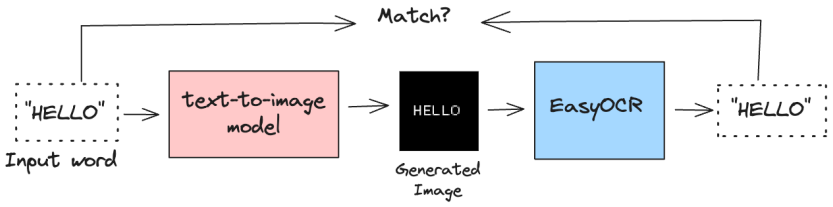


Figure 4.9: OCR Evaluation Process

4.4 Implementation Details

The main Python packages used for all our experiments are *Pillow* (Clark, 2015), *clip-inference* (Beaumont, 2022), *transformers* (Wolf et al., 2020) and *pytorch* (Paszke et al., 2019). An extensive list of all the packages used can be found in the Appendix in Table A.1. The models in our Main Experiments were trained on the NVIDIA T4 GPU with 15GB RAM available for free on the Google Colab [6] environment. The entire code repository for this thesis is publicly available and can be found here: `https://github.com/medha-hegde/master_thesis`.

Our Methodology chapter explained in detail the process used for the three experiment sections: the Preliminary Experiments, the Character Probe Experiment and the Main

[5] https://github.com/JaidedAI/EasyOCR
[6] `https://research.google.com/colaboratory/faq.html`

Experiments. The first two of these are designed to explore the nature of the CLIP text model's "reading" capabilities. The Main Experiments are formulated to expose the most important architectural aspects that improve text rendering in image generation models. This is done by modifying the model architecture and observing its affects on text rendering. In the next chapter, we detail and discuss the results of these experiments.

Chapter 5

Results

The last chapter provided the details of the methods we will use to conduct our analysis of improving text rendering in image generation. We now apply these methods and observe the results of our experiments in this chapter. Following the format of the last chapter, we divide this one into three sections: the Preliminary Experiments, the Character Probe Experiment and the Main Experiments. The results of each section are presented with relevant tables and plots, and their implications are discussed.

5.1 Preliminary Experiments

The results of these four experiments can be seen in Figure 5.1. We use a CLIP score threshold of 0.3 based on the threshold used by the LAION-400M dataset (Schuhmann, 2021) to filter images. This threshold was determined through human evaluations to be a good estimate of semantic image-text matching. The blue bars in Figure 5.1 represent CLIP scores between a corresponding image and text caption, averaged over all 1,000 words. In contrast, the green bars represent the average CLIP scores between an image and all other (remaining 999 words') text captions.

Experiment 1 In this experiment, the CLIP scores are above the threshold value from font size 11 onwards. This means that CLIP is able to recognise that the word is on the image and produces embeddings that represent that word. Scores slightly begin to dip lower at larger font sizes owing to some words being cut-off from the image at those font sizes. For font size 1, where the word cannot be seen on the image, the CLIP scores for both the matching and non-matching words are the same. However, as the font size rises and CLIP is able to recognise the word in the image, the score falls for the non-matching words. This is due to the contrastive pre-training objective of CLIP which maximises the score for a matching image-text pair and minimises it for a non-matching one.

Experiment 2 This experiment demonstrates CLIP's ability to read text in an image even if it is printed on a background image containing a random object that is unrelated to the text itself. This result is in line with those of the typographic attacks evaluated

by Goh et al. (2021). They found that CLIP's multi-modal neurons react strongly to text and this causes a high score for similarity between a text and an image with the text printed in it. All other findings remain the same as in Experiment 1.

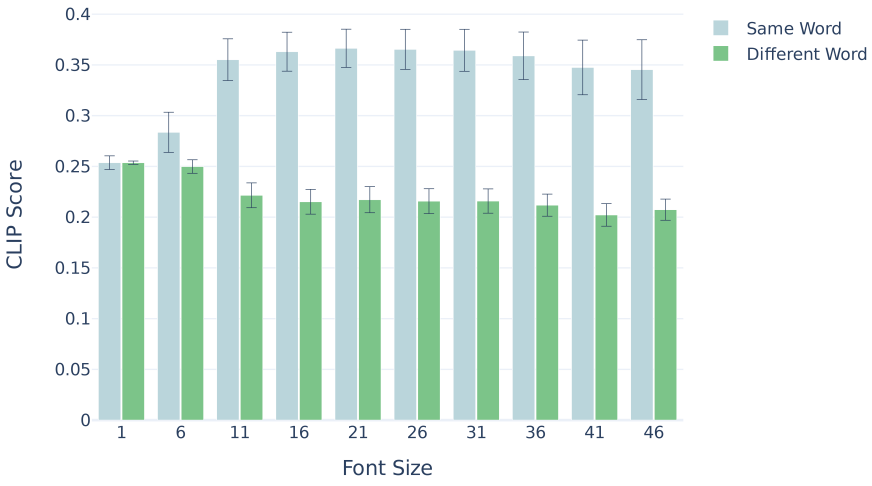
Experiment 3 The CLIP scores shown in Figure 5.1c represent the similarity between a text caption, for e.g., 'an image of the word "HAPPY"', and the 5 corresponding images containing the word "HAPPY" with increasing levels of letter scrambling. The last score, represented by the green bar in the chart, is the average CLIP score between the text caption and all images in the dataset excluding these 5 corresponding images. We consider the CLIP score for 0% scrambling as the baseline to compare the remaining scores to. We observe, as would be expected, decreasing scores for increasing percentages of scrambling. However, the most important finding of this experiment is that even with 100% of the letters in a word being shuffled randomly, its CLIP score is still significantly higher than the average score with other words. This indicates that CLIP is able to somewhat recognise a word in an image even if it is not spelled correctly. It is possible that this phenomenon causes misspellings in text rendered by image generation models when using CLIP embeddings. Specifically, the cause of misspellings involving repeated characters or jumbled letters could be attributable to this. It is also an indication of character-level position encoding not being contained in CLIP embeddings.

Experiment 4 Since these image generation models mostly use CLIP text embeddings for conditioning, we repeat Experiment 3 using only the text embeddings.

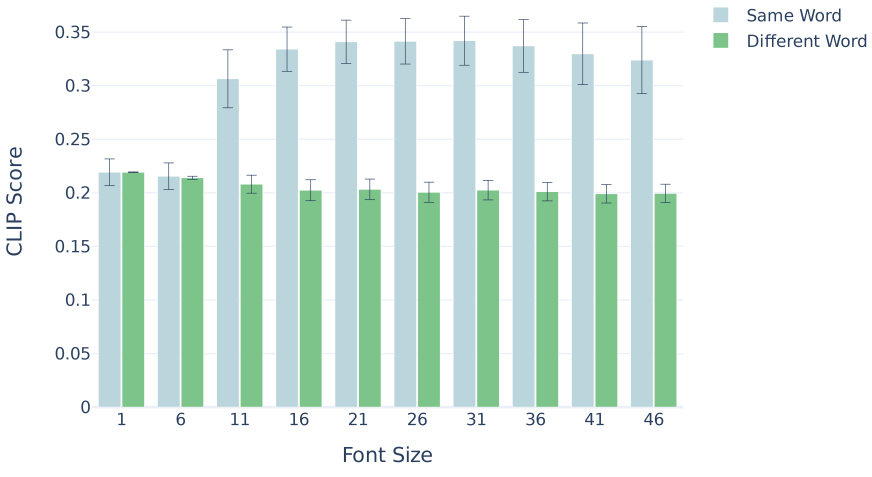
For e.g., we compare the text embedding of the word "HAPPY" with the text embeddings of the words "HPAPY" and "HPPAY" instead of their image embeddings. The first bar in Figure 5.1d with 0% scrambling has a cosine similarity score of 1 (exact match) since it compares a text embedding of a word with itself. The remaining findings remain similar to those of Experiment 3; we see slowly decreasing scores as the level of letter scrambling increases, but still remain higher than the average similarity score with other non-matching words. It should be noted that the difference in scores between the 100% scrambled word text embedding and that of the other words is less significant than in Experiment 3. In Experiment 3, the error bars (representing $\pm$ 1 standard deviation) of the 100% scrambled words do not overlap with those of the non-matching words. This does not remain true for Experiment 4, when comparing text embeddings themselves. This could be another indicator as to the cause of misspelled or gibberish words being rendered during image generation due to spelling information of words not being encoded in the caption text embeddings. The CLIP model's text embeddings do not contain enough information to differentiate significantly between different word spellings.

To further support our findings, we repeated Experiment 4 using the ByT5 text model. We used two variant sizes of the model to observe differences in similarity scores due to model size. As we see in Figure 5.2, both models have lower scores for all scrambling levels as well as for non-matching words. This indicates that CLIP is in fact unable to encode character-level information, possibly due to its tokenization scheme. CLIP utilises BPE tokenization while the ByT5 models are token-free and are byte-level

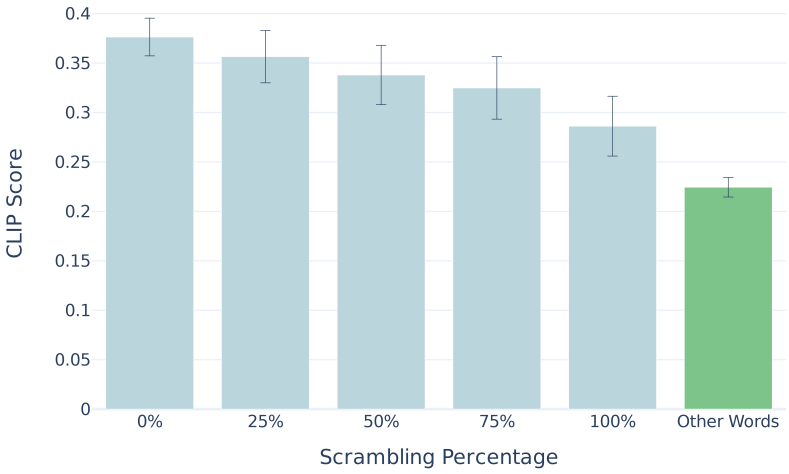
models instead. We will make use of this difference in models in our main experiments in Section 5.3. The larger variant of the ByT5 model has significantly lower scores than its smaller variant, indicating that larger models encode more character level information.



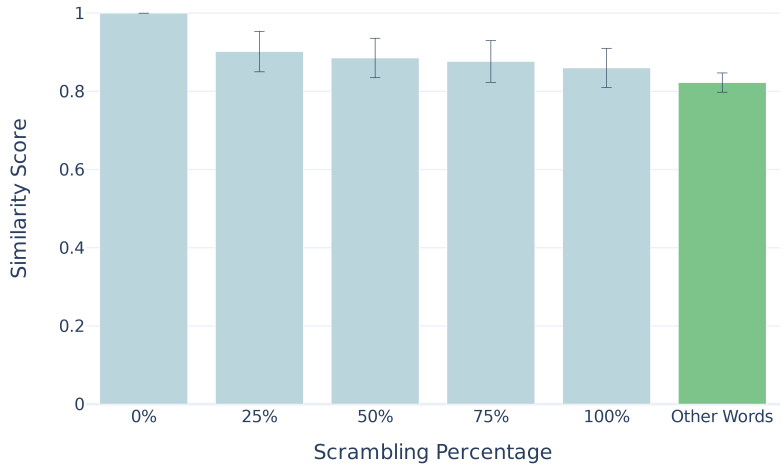
(a) Experiment 1: Increasing Font Size, Blank Background



(b) Experiment 2: Increasing Font Size, Image Background



(c) Experiment 3: Text Scrambling, In Image



(d) Experiment 4: Text Scrambling, Text Only

Figure 5.1: CLIP Experiment Results. Blue bars represent scores of the same words (e.g.: score("happy","hpapy")), while green bars represent scores of different words (e.g.: score("happy","funny")). Error bars represent standard deviation.

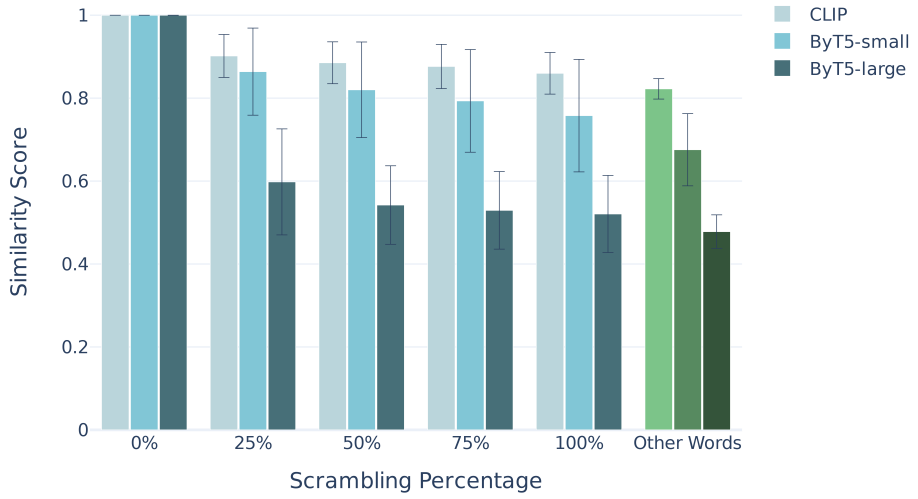


Figure 5.2: Experiment 4 Results for CLIP, ByT5-Small and ByT5-Large . Blue bars represent similarity scores of the same words (e.g.: score("happy","hpapy")), while green bars represent similarity scores of different words (e.g.: score("happy","funny")). Error bars represent standard deviation.

5.2 Character Probe Experiment

Model	# Parameters	Pre-Training Objective	F1-score
GPT-J	6B	Next Word Prediction	88.46
GPT-2	124M	Next Word Prediction	78.21
Roberta	125M	Masked Language Modeling (MLM)	80.61
BERT	110M	MLM, Next Sentence Prediction	69.6
CLIP-large	123M	Image-text Alignment	**87.35**
CLIP-base	63M	Image-text Alignment	**87.89**
T5-large	770M	MLM	**84.34**
T5-base	220M	MLM	**82.15**

Table 5.1: **Character Probe Experiment Results**. Models with F1-scores in bold are our additions to the list of models analysed by Kaushal and Mahowald (2022).

We see in Table 5.1 that the CLIP models perform very well on this task. They have a higher F1-score than the other language models tested by Kaushal and Mahowald (2022) with the exception of GPT-J, which is a 6 billion parameter model. The CLIP-

base text model, in contrast, only has 63 million parameters but is able to outperform the other larger language models listed in the table. This finding goes against those of Kaushal and Mahowald (2022) where they suggest that larger models perform better.

Embedding Type Our findings also suggest that CLIP does in fact contain character-level information in its frozen token embeddings. This may appear contrastive to the hypothesis from Section 5.1 that CLIP embeddings do not contain character information. But it is important to differentiate between a model's frozen token embeddings and its final text embeddings. The latter is what is used by image generation models for conditioning while the former is what CLIP itself trains on during pre-training. The final embeddings are obtained after passing the token embeddings through the entire transformer architecture of the CLIP text encoder model.

Pre-Training Objective Another reason for CLIP's unexpectedly high score could be due to differing pre-training objectives compared to the other language models listed. As mentioned in Section 2.5, CLIP is trained with the objective of image-text alignment to be able to represent an image's visual elements within the text embeddings. Since CLIP has been shown to demonstrate OCR capabilities, it can be suggested that it is more important for it know the spelling of words that appear in an image. On the other hand, the other language models in the table are all pure language models and have the pre-training objectives of masked language modelling or next-word prediction. They are not necessarily required to contain character information in their embeddings since they primarily work on tokens.

5.3 Main Experiments

5.3.1 Text Encoder Type

The first architectural aspect of image generation we test is the text encoder type. We compare the model's performance with three different text encoders - CLIP, T5 and ByT5. As we can see in Figure 5.3, the ByT5 model consistently outperforms the other two models in every frequency bucket by a large margin. The T5 model also scores higher than the CLIP model in all buckets, although by a smaller margin. This is in line with the findings of Saharia et al. (2022), where they compare the CLIP and T5 models as conditioning models in image generation and concluded that T5 is preferred especially for "ability to generate quoted text". However, it is important to note that the T5 variant used by Saharia et al. (2022) was the largest variant T5-XXL model which has 11 billion parameters. Our model on the other hand is using the T5-Small variant with only 60 million parameters, which may be the cause of the smaller difference in performance between CLIP and T5 in our experiment. We explore the effect of the number of parameters in the text model in a later section.

Both CLIP and T5 perform worse in the 20-30% and Bottom 50% frequency buckets compared to the other buckets. This could be due to both models employing subword-based tokenizers which employ tokenization algorithms that depend on the frequency of words in their respective training corpora. The ByT5 model on the other hand, just uses the raw UTF-8 bytes of the characters and hence it still shows a 23% exact match score for the Bottom 50% bucket, compared to the lower 5.6% and 0.23% scores for the T5 and CLIP models respectively. This significant difference in performance for the ByT5 model in all buckets is in agreement with the main findings of Liu et al. (2022) where the researchers demonstrated the ability of the ByT5 model to spell more accurately than the T5 model. While they compared the largest variants of both models, our experiment with the smallest variants still reach the same conclusions. The results indicate that using a character-aware model is conducive to more accurate text rendering in diffusion models.

5.3.2 Text Encoder Size

In Figure 5.4, we observe the results of the models with larger variants of the three text encoders. We see a similar pattern as in Figure 5.3 with the ByT5 model outperforming the T5 and CLIP models. However, a marked difference in this case is the gap between the T5 and CLIP models. The T5 model greatly outperforms the CLIP model, with scores closer to the ByT5 model, in contrast with the scores of the smaller T5 variant. This indicates that both T5 and ByT5 contain more information in their embeddings that aid the image generation model to more accurately render text in images. It also indicates that scaling up the pre-trained text encoder without scaling up the image generation model itself can improve text rendering. This is also supported by the findings of Saharia et al. (2022) where they compare all 5 variant sizes (ranging form 60 million to 11 billion parameters) of the T5 models. They concluded that scaling up the size of the text encoder resulted in increased image-text alignment, with the largest T5-XXL-

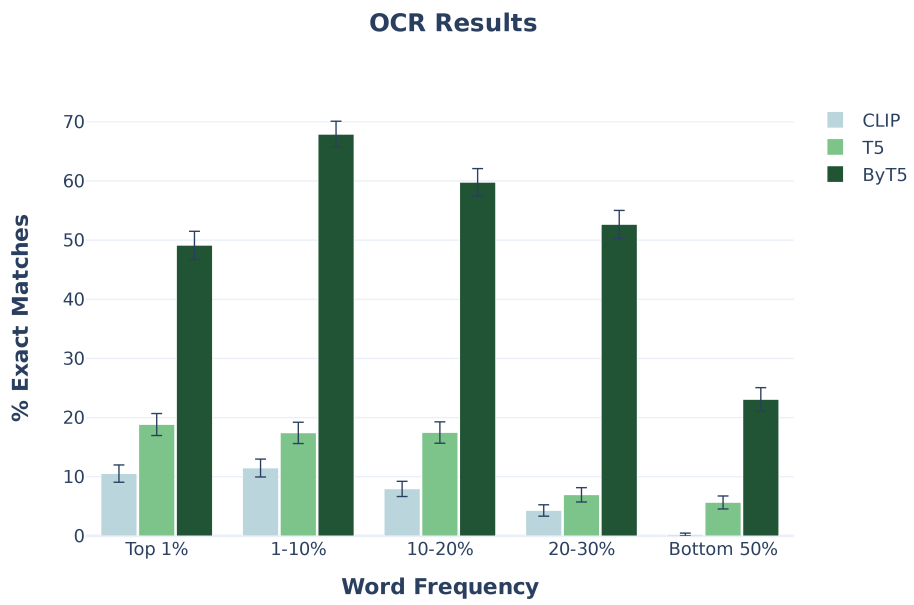


Figure 5.3: Text Encoder Type Experiment Results

conditioned model having the best CLIP scores.

On the other hand, the CLIP model does not seem to be similarly benefiting from an increase in number of parameters. In fact, the CLIP-Large model performs worse than the CLIP-Small model, as seen in Figure 5.5. A possible explanation is that when scaling up the CLIP model, the text encoder was only scaled by width and not by depth. The other two models are scaled both by width and depth, by adding additional layers, while CLIP-Large has the same number of layers as the CLIP-Small model. Tay et al. (2021) suggest that scaling depth and not width is more efficient when it comes to transformer model scaling. This could be the cause of reduced scores for the CLIP-Large variant with regard to conditioning during image generation. It is important to note here that both Stable Diffusion (Rombach et al., 2022) and DALL-E 2 (Ramesh et al., 2022) use the larger variant of CLIP.

In the word frequency bucket breakdown in Figure 5.6, we see that the Large ByT5 scores more than twice the score of the Small ByT5 model in the Bottom 50% bucket. Similarly, the Large T5 model shows significant improvements in scores over its small T5 model in all buckets. We can extrapolate that by using even larger variants of both models, like the ByT5-XXL and the T5-XXL, we would observe still higher scores in this task.

5.3.3 Training Time

Next, we scale the CLIP ViT-B/32-conditioned U-Net model by training it for an increased number of training steps. Looking at Figure 5.7, training for just an additional

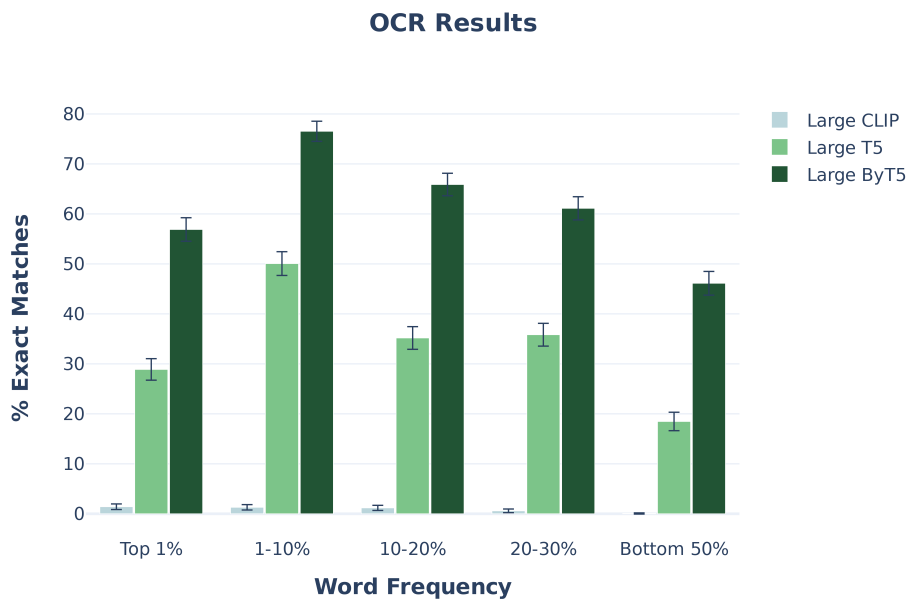


Figure 5.4: Text Encoder Size Experiment Results

100 epochs results in doubling of the OCR score from 7% to 14%. With just these additional 100 epochs of training, the CLIP model is able to match the score of the T5-Small model. With subsequent additional training steps, the score does continue to slowly rise, but eventually starts to fall around epoch 750. From this experiment, we can conclude that while some additional training steps results in an increase in performance for the CLIP model, this increase is not linear and is still substantially smaller than that of the ByT5 model.

5.3.4 Combined Embeddings

Finally, in order to leverage the text rendering advantages of the ByT5 model while keeping the other general image generation capabilities of the CLIP and T5 models, we run the experiment with combined text embeddings. In Figure 5.9, we see that the combined T5-ByT5 model has the highest score in all frequency buckets. The other two combinations, i.e., the CLIP-T5 and the CLIP-ByT5 display similar performance to each other. When compared to the previous models without combined embeddings (Figure 5.10), we see that the two CLIP-combined models are similar in performance to the T5 Small model alone. This shows that combining the CLIP model with a pure language model can result in an increase in text rendering capability. From previous results, we can also hypothesise that these gains would be larger with larger variants of the T5 and ByT5 models.

For the T5-ByT5 combined model, the increase in performance over the pure T5 model is tripled when combined with the ByT5 model and almost matches the perfor-

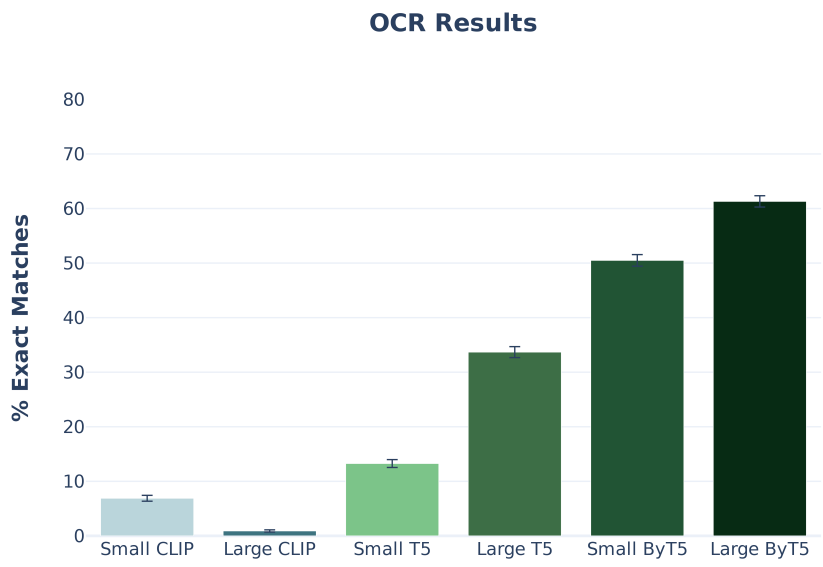


Figure 5.5: Text Encoder Size Experiment Results, Large vs Small

mance of the ByT5 model itself. This implies that the ByT5 model has a multiplicative impact when combined with the T5 model while this impact is much less significant when combined with the CLIP model. The text rendering capabilities of the combined T5-ByT5 embeddings has also been explored by Liu et al. (2022), where they combined the T5-XXL model embeddings with the ByT5-small model. They observed that this combined model retains the image-text alignment level of the T5 model, while improving its text rendering capability owing to the ByT5 model.

In Figure 5.11, we can observe the breakdown of results by word frequency bucket along with the pure (small) CLIP, T5 and ByT5 models. The T5-ByT5 model is consistently close to the scores of the ByT5 model in all buckets. Interestingly, the CLIP-T5 model is able to score higher than the CLIP and T5 models alone in the 1-10% and the 20-30% buckets.

In this chapter, we presented the results of our experiments and discussed their implications with respect to our research topic. The Preliminary Experiments indicate that the CLIP text model can perform OCR (optical character recognition), but it struggles to identify character positions within words. The Character Probe model reveals an interesting discovery: the token embeddings of CLIP do contain character-level information. The Main Experiments show that replacing the CLIP text encoder with the character-level large language model ByT5 leads to a significant enhancement in the text rendering abilities of the image model. Furthermore, it is observed that using larger text models, extending the training time, and combining character-level text embeddings yield further

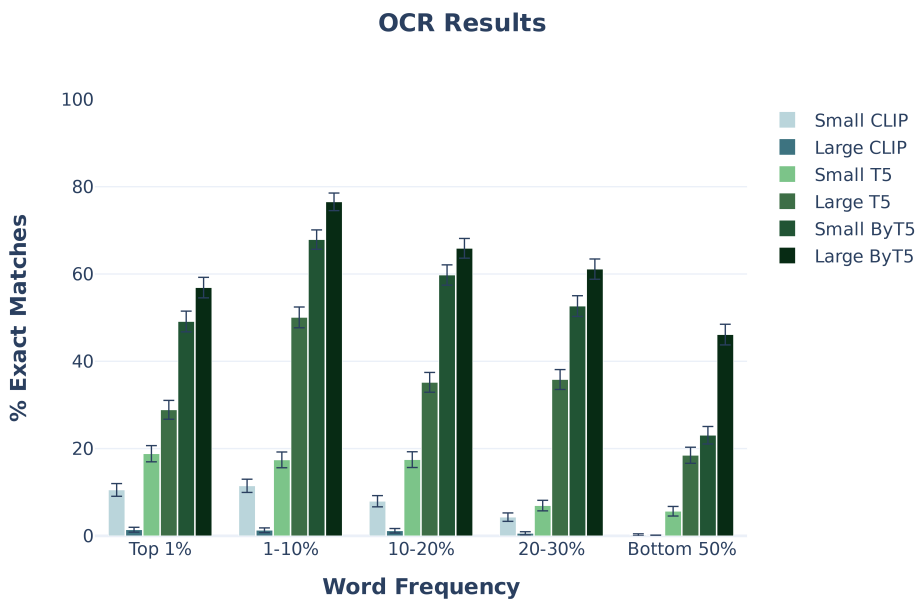


Figure 5.6: Text Encoder Size Experiment Results, by Word Frequency Bucket

improvements in performance. We move on to the Conclusion of our analysis in the next chapter, where we also detail the limitations of our analysis and the potential for future research.

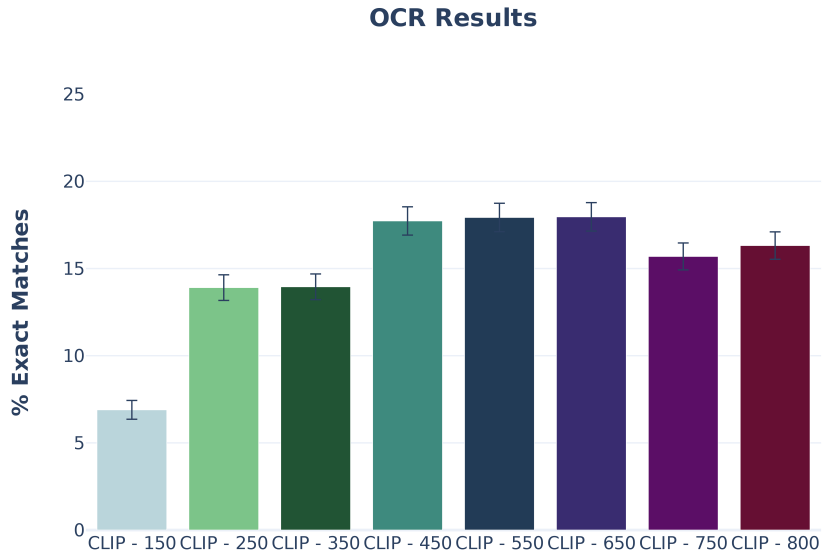


Figure 5.7: Training Time Experiment Results. CLIP-150 is the model trained for 150 epochs with the CLIP ViT-B/32 model. Remaining models are evaluated at epochs specified in the model name.

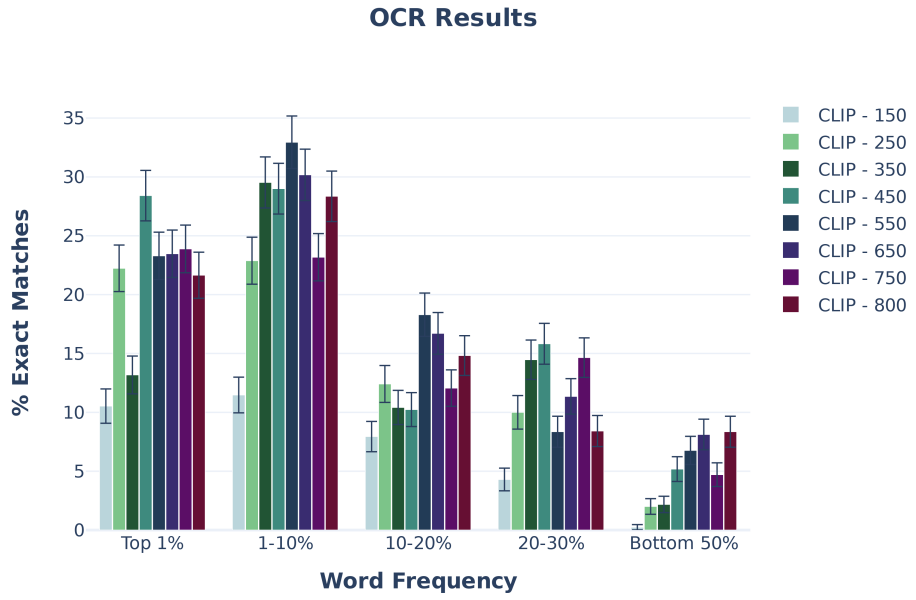


Figure 5.8: Training Time Experiment Results, by Word Frequency Bucket

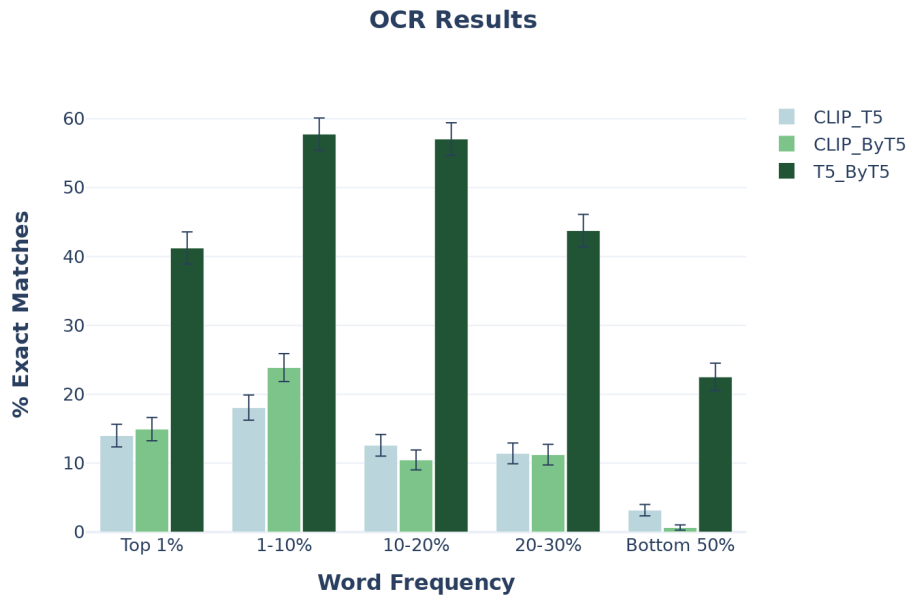


Figure 5.9: Combined Embeddings Experiment Results

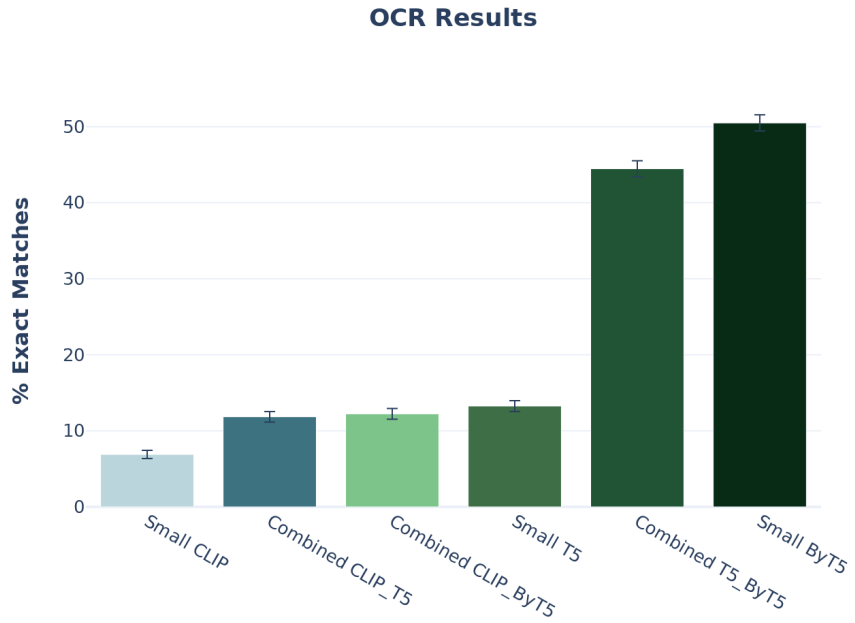


Figure 5.10: Combined Embeddings Experiment Results, Comparison with CLIP, T5 and ByT5 models.

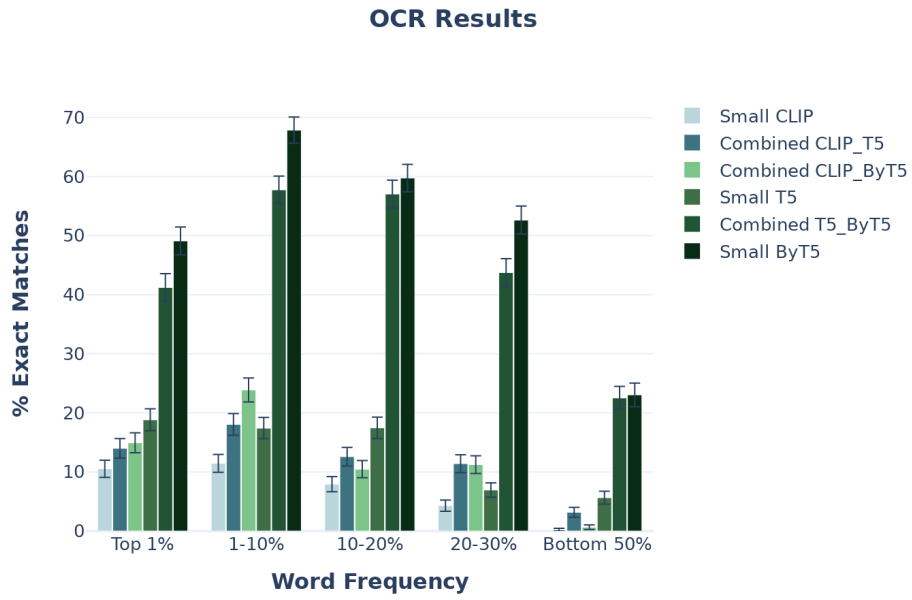


Figure 5.11: Combined Embeddings Experiment Results, Comparison with CLIP, T5 and ByT5 models, by Word Frequency Bucket

Chapter 6

Conclusion

The previous chapter spelled out the results and the interpretation of our experiments. With this final chapter, we conclude our findings, list the limitations and the possibilities for further research.

Our first Research Question asks *"What are the causes for incorrect text rendering in current image generation models?"*. We sought to answer this by first understanding the nature of the spelling capabilities of the CLIP model using the Preliminary Experiments (Section 5.1) and the Character Probe Experiment (Section 5.2). We found that the CLIP model requires a certain minimum text size for the text to be recognised by it and be encoded in its embeddings, regardless of the background image. The preliminary experiments also suggest that CLIP does not encode character level information in its embeddings. These findings point to possible causes of incorrect text rendering in image generation models that use the CLIP model for text-conditioning. The results from the Character Probe Experiment, on the other hand, do not provide an answer to this research question due to the usage of token embeddings instead of text embeddings. Instead, this experiment revealed an unexpected characteristic CLIP's token embeddings: enough character information is embedded in them to be useful in a character-classification task. While this could be attributed to CLIP's unique OCR capabilities, it is not objectively clear why CLIP is able to match the performance of large language models.

The Main Experiments (Section 5.3) seek to simultaneously answer all three Research Questions by providing multiple possible causes of incorrect text rendering and providing modifications to improve text rendering by training a diffusion model from scratch. We do this by reducing the scope of our experiments to the simplest form to focus only on text rendering capabilities of image generation models. On this simplified task, our base model uses the CLIP-base text encoder for conditioning the 64x64 U-Net model for 150 epochs. We consider a modification to this base model to be an "improvement" if the score of the model as measured by the OCR evaluation rises. The modifications that improved text rendering were: 1) using a pre-trained large language model like T5 or ByT5 instead of CLIP, 2) increasing training time of the base model and , 3) including character-level information by combining text embeddings with those of a character-level text encoder like ByT5.

Our main conclusion from all experiments is that the CLIP model is a significant source of incorrect text rendering. This is possibly attributable to its pre-training objective, its smaller model size and its tokenization scheme. Our findings are in line with those of Liu et al. (2022), who performed a similar analysis on a much larger, more general dataset of 400 million images. They compared only two language models: the T5 and the ByT5, while we also included CLIP. We trained on a small, uniform dataset along with a smaller image generation model that required significantly less compute. Despite this, we were able to reach the same conclusions as Liu et al. (2022). Specifically, our analysis and theirs both conclude that the ByT5 model, owing to being a tokenizer-free encoder, is better than a tokenizer-based model like T5 or CLIP, for text rendering as evaluated by OCR models.

6.1 Limitations

Small Scope The scope of our main experiments is limited to the specific type of image we chose: a 64x64 plain black image with upper-case text printed in white in the centre of the image. The results of our study can only be applicable to this very specific set-up. In reality, images like this are rarely encountered and hence it is possible that the findings of this study don't generalise to other types of images containing text. Variations in font size, colour, style and text position, among others, were not taken into consideration. Scripts in other languages that use different characters as well as Latin scripts that use accented characters were also not within our scope.

Implication for Non-text Images Our conclusions are only in the context of the one task of text rendering in images. However, the current state-of-the-art image generation models are very general and are trained to generate images of almost any category. The reason CLIP is used in current image generation models (Ramesh et al., 2022; Rombach et al., 2022) is that it is able to capture visual information in its text embeddings, allowing the image generation models to successfully generate high fidelity images. It is possible that this capability may be lost when replacing the CLIP text encoder as a modification for improved text rendering.

Compute Constraints Our study was limited to the minimum model size, training time and dataset size that was possible to train on the NVIDIA T4 GPU with 15GB RAM available for free on the Google Colab [1] environment. There were usage limits due to which training would be crashed or time limits would be enforced. Access to more compute would have accelerated speed of training our models, thus enabling the possibility of conducting more experiments. It would have also allowed for the training of larger models and the usage of larger text models such as T5-XXL or ByT5-XXL to observe the affect of this on the text rendering capabilities.

Other Causes We focused mainly on the text encoder used for conditioning of image generation models, but it is possible there are other model aspects we did not investigate

[1] https://research.google.com/colaboratory/faq.html

that could have an impact on text rendering. Stable Diffusion (Rombach et al., 2022) uses latent diffusion by first reducing the image to a latent space and then running the diffusion process. We did not include this latent process in our analysis but it could be a another contributor to incorrect text rendering since there is information loss (Rombach et al., 2022, Section 4.1) during compression of the images. Additionally, an investigation of the image-text datasets used by image generation models was not conducted. It is possible that the representation of text in images was comparatively low and this was another cause of incorrect text rendering.

6.2 Risks

While our analysis does not immediately pose such risks, there is potential for it to aid in the development of image generation models that can create images with fewer markers of being AI-generated. It is therefore important to mention the risks of current image generation models due to their potential for misuse via the generation of harmful or misleading content. Mishkin et al. (2022) list some of these risks that include illegal content, bias, exploitation, disinformation and copyright issues, among many others.

6.3 Future Work

Building on our work, there are various aspects of text rendering that can be explored in future work. It would be interesting to see if the application of our methodology on multilingual datasets involving languages other than English would result in the same conclusions. The incorporation of the latent diffusion in the image generation architecture would allow for a more representative modelling of current image generation models. As mentioned in Section 6.1, using a training dataset of different font styles, font sizes, colours, etc would allow for a more diverse and "realistic" dataset. An even more realistic dataset could involve the collection of natural images with text in the image, similar to the IIIT 5K-word dataset (Mishra et al., 2012). With regards to the training set-up of our diffusion model, different sampling strategies (Karras et al., 2022) or different conditioning mechanisms (Yang et al., 2022, Section 7.1.1) may produce different text rendering results.

We conclude this Thesis by connecting our results to the Research Questions. We found some probable causes of incorrect text rendering and also provided possible solutions to this research problem. Some limitations include the scope of our experiments, the applicability of our modifications for non-text images and compute constraints. Finally, we discussed the risks and potential for further research in this domain. The proceeding Appendix chapter contains additional information that provides clarity to our work presented in previous chapters.

Appendices

Appendix A

Appendix

A.1 List of Python Libraries Used

Library	Description
datasets	Datasets and evaluation metrics for natural language processing
nltk	Standard Library for NLP tasks
random	Pseudo-random number generators for various distributions
json	Library to work with json files
regex	Library to work with regular expressions
os	Library to work with the file system of an operating system
Pillow	Image processing capabilities
clip-inference	Library to generate CLIP embeddings
numpy	Mathematical operations on arrays
pandas	Operations on tabular data
scikit-learn	Predictive data analysis tools
plotly	Interactive Plots
matplotlib	Plotting capabilities
requests	Make HTTP requests
torch	Machine Learning Framework
transformers	Allow for usage of pre-trained ML models
einops	Enables tensor operations
wandb	Tracking model hyperparameters, metrics and outputs
easyocr	Multilingual OCR module

Table A.1: List of Python libraries used.

A.2 Preliminary Experiments Sample Images

Figure A.1: Experiment 1: Increasing Font Size, Blank Background

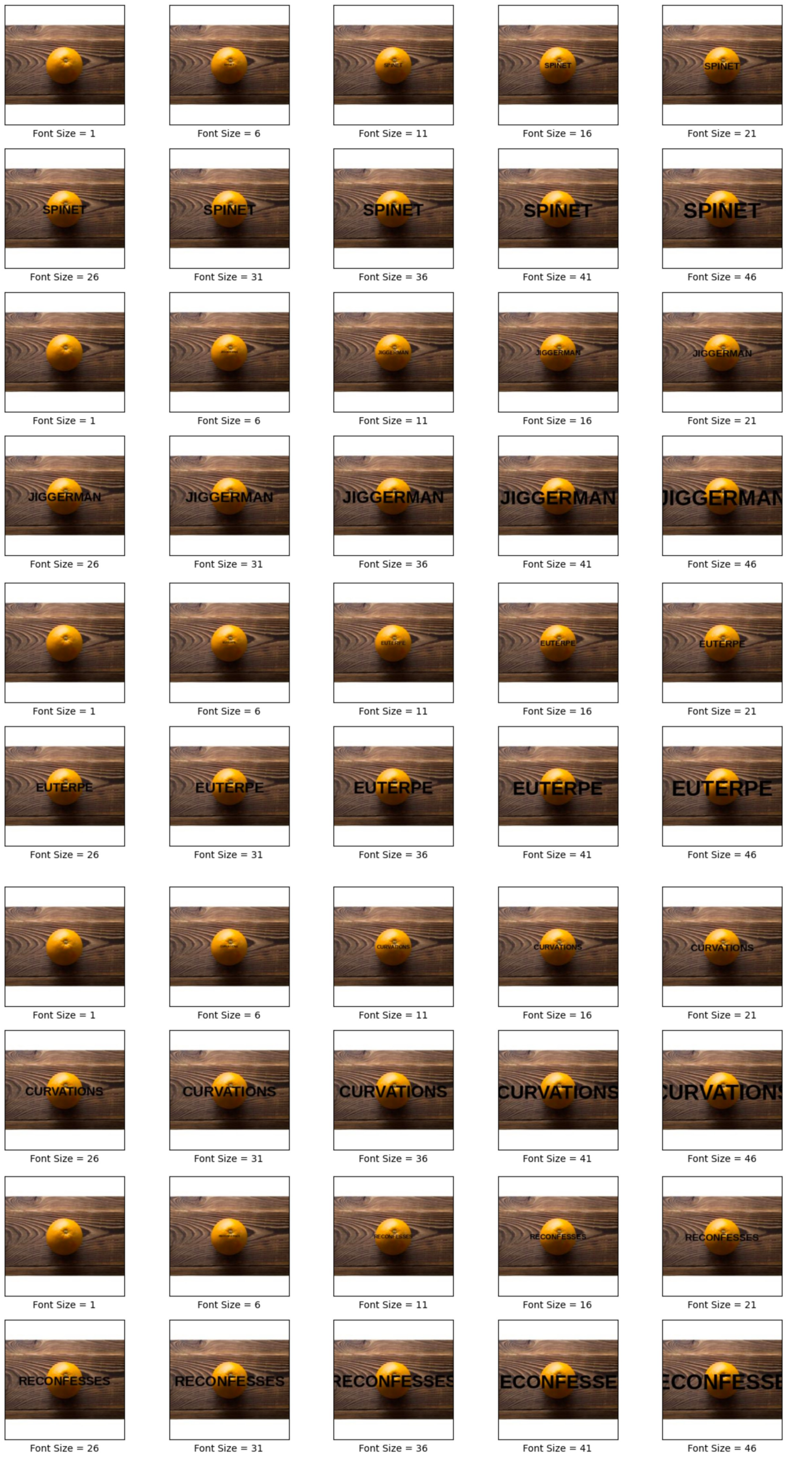


Figure A.2: Experiment 2: Increasing Font Size, Image Background

Figure A.3: Experiment 3: Text Scrambling, In Image

A.3 Character Probe Experiment Sample Dataset

Below is a part of the token dataset created for the Character Probe Experiment, for the letter "a". The format of each item is as follows: ["token",token ID, binary label if letter is contained in token]. The </w> in the tokens represents white-space, indicating that the original word has been split during tokenization.


```
   "a":  [[["bribe</w>", 40042, 0], ["institutional</w>", 27442,
1], ["dancers</w>", 13153, 1], ["shill</w>", 30090, 0], ["khi",
39062, 0], ["anime", 23314, 1], ["incomplete</w>", 34040, 0], ["dis-
per", 34499, 0], ["aoa</w>", 47119, 1], ["gaa</w>", 10715, 1],
["vism</w>", 15514, 0], ["lola</w>", 19065, 1], ["slows</w>", 46855,
0], ["divya</w>", 47767, 1], ["vian", 40487, 1], ["dublin", 20707,
0], ["comedian</w>", 13848, 1], ["darshan</w>", 34564, 1], ["grady</w>",
33980, 1], ["atta", 7282, 1], ["consor", 27108, 0], ["pope", 16994,
0], ["gustav", 32062, 1], ["ritos</w>", 33507, 0], ["spoiled</w>",
21399, 0], ["search</w>", 1920, 1], ["moment", 12197, 0], ["cos",
5865, 0], ["marmalade</w>", 39068, 1], ["las</w>", 2552, 1], ["cone",
39279, 0], ["nagar</w>", 17490, 1], ["relationship", 47239, 1],
["hle</w>", 32389, 0], ["libertarian</w>", 35067, 1], ["david-
bowie</w>", 44448, 1], ["sapp</w>", 40729, 1], ["patty", 48741,
1], ["kja", 41048, 1], ["bloody</w>", 9468, 0], ["uta</w>", 25925,
1], ["desper", 10144, 0], ["feather</w>", 17871, 1], ["burns</w>",
10234, 0], ["kicked</w>", 12479, 0], ["mcnamara</w>", 37506, 1],
["produ", 27852, 0], ["peng</w>", 39200, 0], ["sally</w>", 13349,
1], ["rosso</w>", 33023, 0], ["sacrific", 16919, 1], ["ghi", 22436,
0], ["mine</w>", 3347, 0], ["nga</w>", 35477, 1], ["ished</w>",
35341, 0], ["mash", 12317, 1], ["stops</w>", 9822, 0], ["fees</w>",
11765, 0], ["orities</w>", 7903, 0], ["scand", 8110, 1], ["in-
tentionally</w>", 31215, 1], ["ul", 914, 0], ["broken", 26126,
0], ["receivers</w>", 45294, 0], ["cancel</w>", 21546, 1], ["moro</w>",
31613, 0], ["blanks</w>", 48599, 1], ["commute</w>", 15898, 0],
["discover", 10539, 0], ["shore", 14717, 0], ["match", 7733, 1],
["flats</w>", 17228, 1], ["obe", 11897, 0], ["tavern</w>", 18644,
1], ["parental</w>", 28339, 1], ["pierce", 48462, 0], ["beirut</w>",
22352, 0], ["stonehenge</w>", 42417, 0], ["matory</w>", 25519,
1], ["bringback", 28969, 1], ["uneven</w>", 43204, 0], ["paddle</w>",
20811, 1], ["aaj", 28727, 1], ["screenshots</w>", 26783, 0], ["urer</w>",
20864, 0], ["enty</w>", 5657, 0], ["pleasures</w>", 29513, 1],
["chapter</w>", 6014, 1], ["choic", 35606, 0], ["asics</w>", 31097,
1], ["ask</w>", 2765, 1], ["sher</w>", 4510, 0], ["reina</w>",
43847, 1], ["ica</w>", 2576, 1], ["smartwatch</w>", 35798, 1]]]
```

A.4 Sample Words from each Word Bucket

Random sample of words from each Word Bucket as detailed in Section 4.3.2. 30 words are sampled randomly for each bucket, excluding the Top 1-20% and the Top 20-30% which have fewer words to sample from. They contain 22 and 5 words respectively.

Top 1%: `['rule', 'easily', 'members', 'door', 'superior', 'actually', 'future', 'lies', 'sat', 'map', 'ray', 'affairs', 'baking', 'prepared', 'install', '1', 'covered', 'focus', 'bird', 'club', 'warm', 'injuries', 'family', 'bad', 'seeing', 'holy', 'expect', 'draft', 'some', 'stations']`

Top 1-10%: `['cops', 'broadway', 'textures', 'tweet', 'rack', 'revolt', 'lakes', 'grove', 'fluffy', 'carrier', 'geller', 'dancer', 'hygiene', 'educator', 'intuit', 'fondue', 'cognac', 'arlene', 'borrower', 'sonny', 'standoff', 'fume', 'airtight', 'discern', 'tinker', 'roughly', 'pied', 'wring', 'consular', 'entree']`

Top 10-20%: `['fardh', 'lmao', 'divulged', 'mandurah', 'glick', 'hillock', 'oif', 'elope', 'freeland', 'yous', 'ranchero', 'belles', 'glitchy', 'moldable', 'tnx', 'lupo', 'bew', 'solanas', 'beps', 'wex', 'ramayana', 'runout']`

Top 20-30%: `['tetras', 'halbert', 'pennisi', 'cleated', 'phthalo']`

Bottom 50%: `['folino', 'dalleys', 'babesias', 'bone', 'bedsock', 'cyllene', 'agleted', 'overburn', 'unduped', 'saphers', 'inblands', 'chasids', 'porage', 'lexiteer', 'serja', 'begohm', 'cheezes', 'rampier', 'bedumbs', 'crumcake', 'nangca', 'rishitin', 'expulses', 'baroflex', 'preferer', 'exeme', 'sniders', 'solands', 'besteals', 'uresis']`

A.5 Main Experiments Training Process

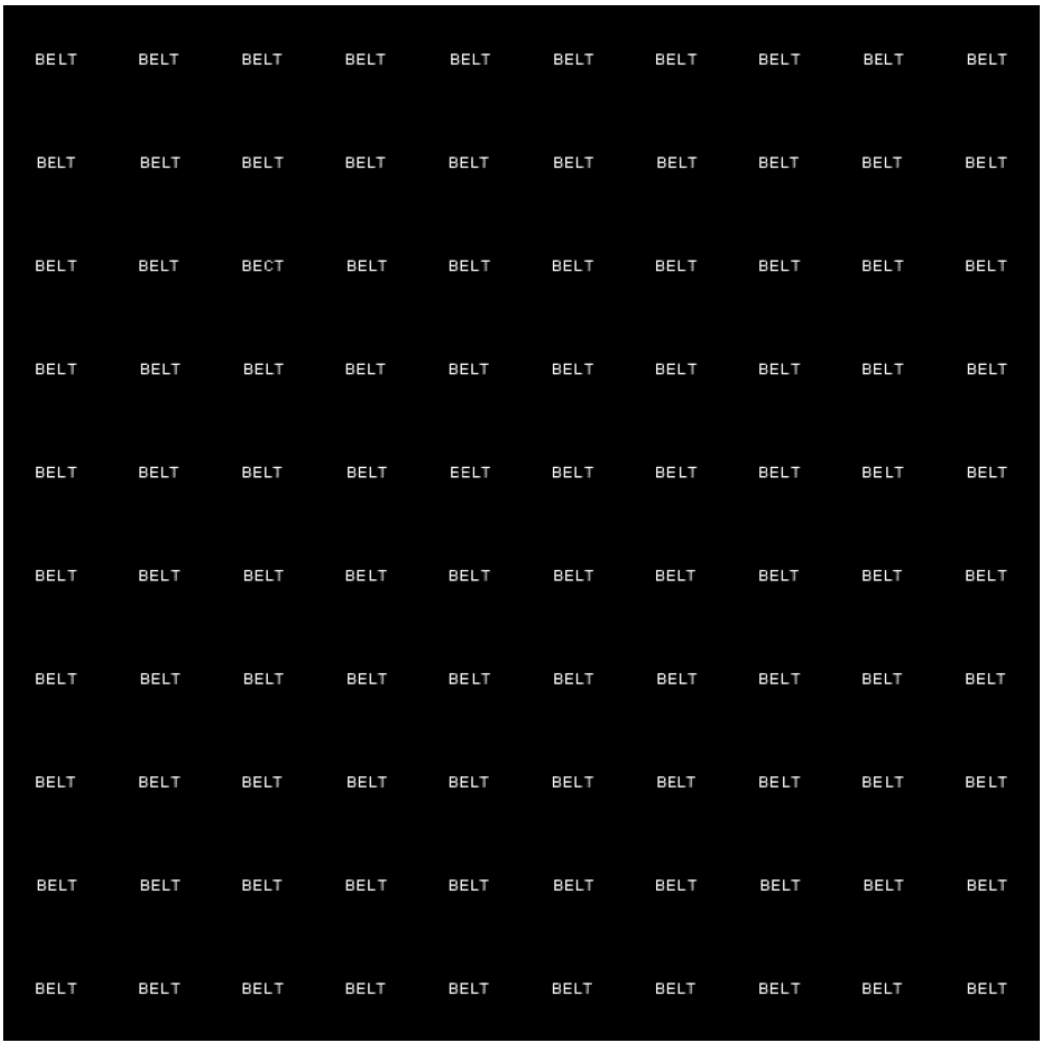

Epoch 49
TECGL
Epoch 50
HLCHO
Epoch 51
IHDLEL
Epoch 52
IBEIHL
Epoch 53
IELCIIL
Epoch 54
VEOLIL
Epoch 55
KELLIB
Epoch 56
REUKL
Epoch 57
HCINE
Epoch 58
HELLL
Epoch 59
RELLL
Epoch 60
HLEIDEL
Epoch 61
MLCE
Epoch 62
MLELIN
Epoch 63
HELLL
Epoch 64
HDELH
Epoch 65
HIELIS
Epoch 66
HEEDUL
Epoch 67
I ELILC
Epoch 68
IBEKCH
Epoch 69
HEUND
Epoch 70
HELCH
Epoch 71
FEOUU
Epoch 72
HEEICL
Epoch 73
HELUHI
Epoch 74
HEBEL
Epoch 75
HELLD
Epoch 76
HIELLO
Epoch 77
HAEEU
Epoch 78
HIELD
Epoch 79
NELIEL
Epoch 80
HEEUKI
Epoch 81
HELDD
Epoch 82
HIEI IE
Epoch 83
WELLE
Epoch 84
HEOIE
Epoch 85
HIEELL
Epoch 86
FELEUE
Epoch 87
WELLU
Epoch 88
WLEOL
Epoch 89
HELUE
Epoch 90
HEEUIC
Epoch 91
HELLIL
Epoch 92
HEILLL
Epoch 93
HELUID
Epoch 94
HE IELD
Epoch 95
NEOD
Epoch 96
HELUIE
Epoch 97
HEHO
Epoch 98
HELBLC
Epoch 99
HEELO
Epoch 100
FELUIC
Epoch 101
WEELO
Epoch 102
HELCL
Epoch 103
HELILLI
Epoch 104
HUEUC
Epoch 105
HELUE
Epoch 106
HELLCK
Epoch 107
NIGLE
Epoch 108
HEOIO

Figure A.4: Model with CLIP ViT-B/32 Training Process Visualised. Images generated with the prompt *"a black and white image of the word "HELLO""*, sampled after each epoch.

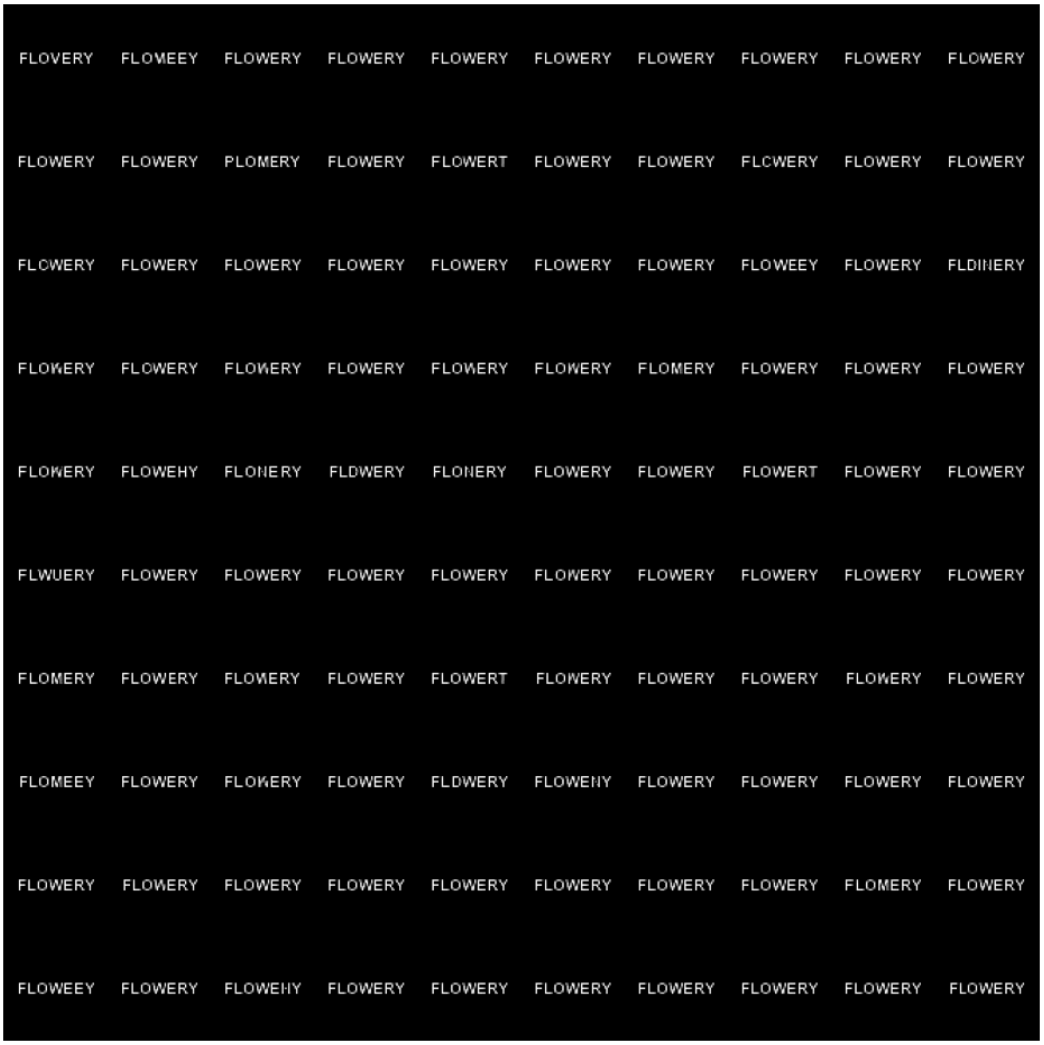
A.6 Sampled Images after Training

The images below are the samples generated by the ByT5-Large model. One word is sampled from each of the word frequency buckets and 100 images are generated using this word.

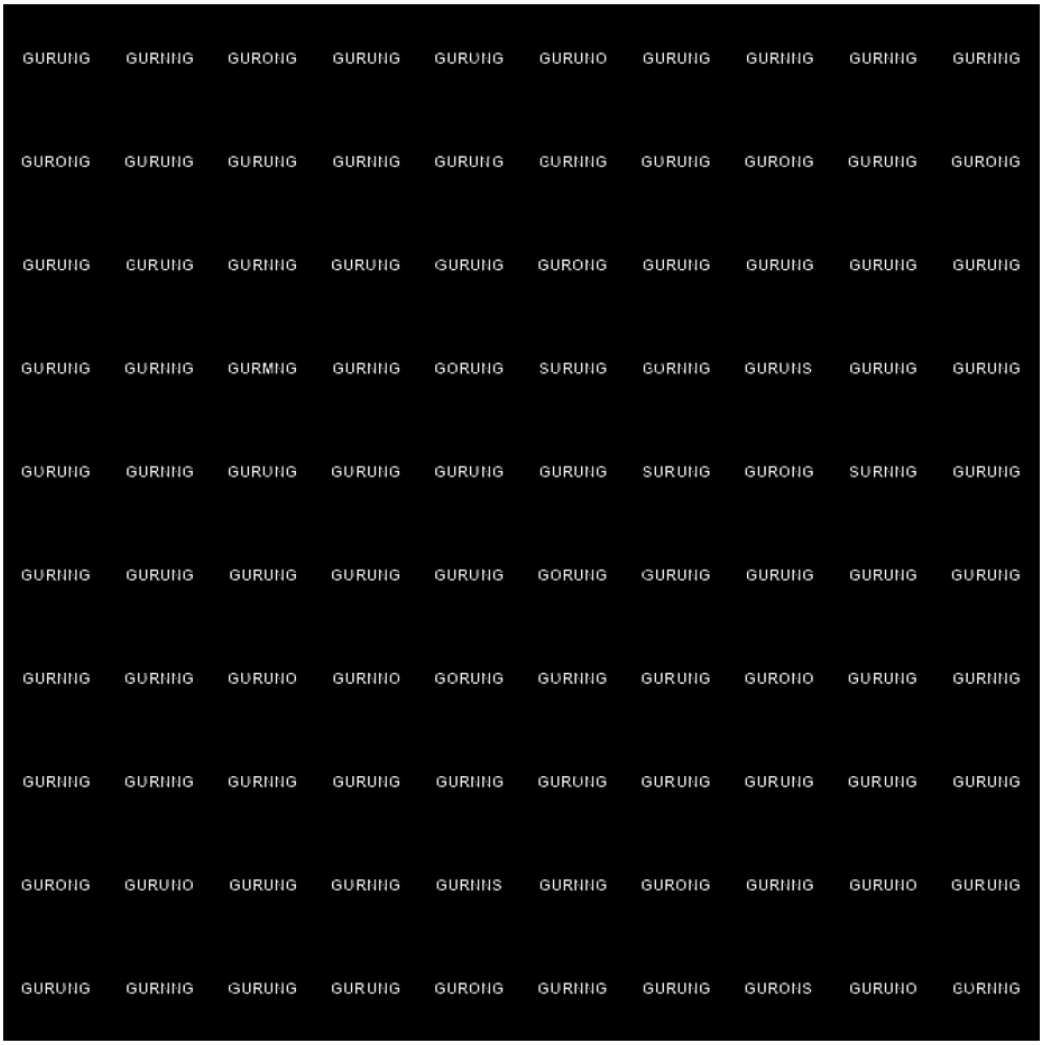
Caption: "a black and white image of the word "BELT"
OCR score: 98%



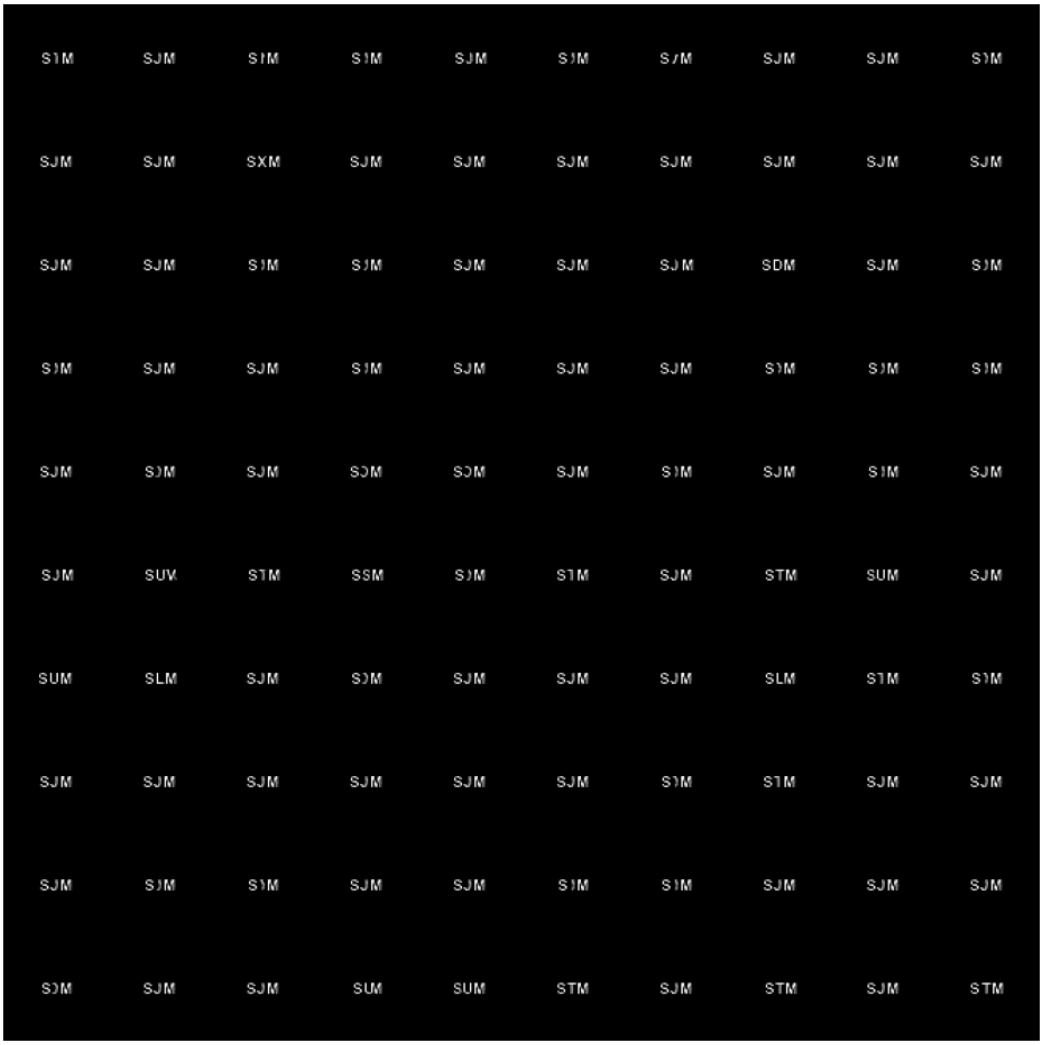
Caption: "a black and white image of the word "FLOWERY"
OCR score: 66%



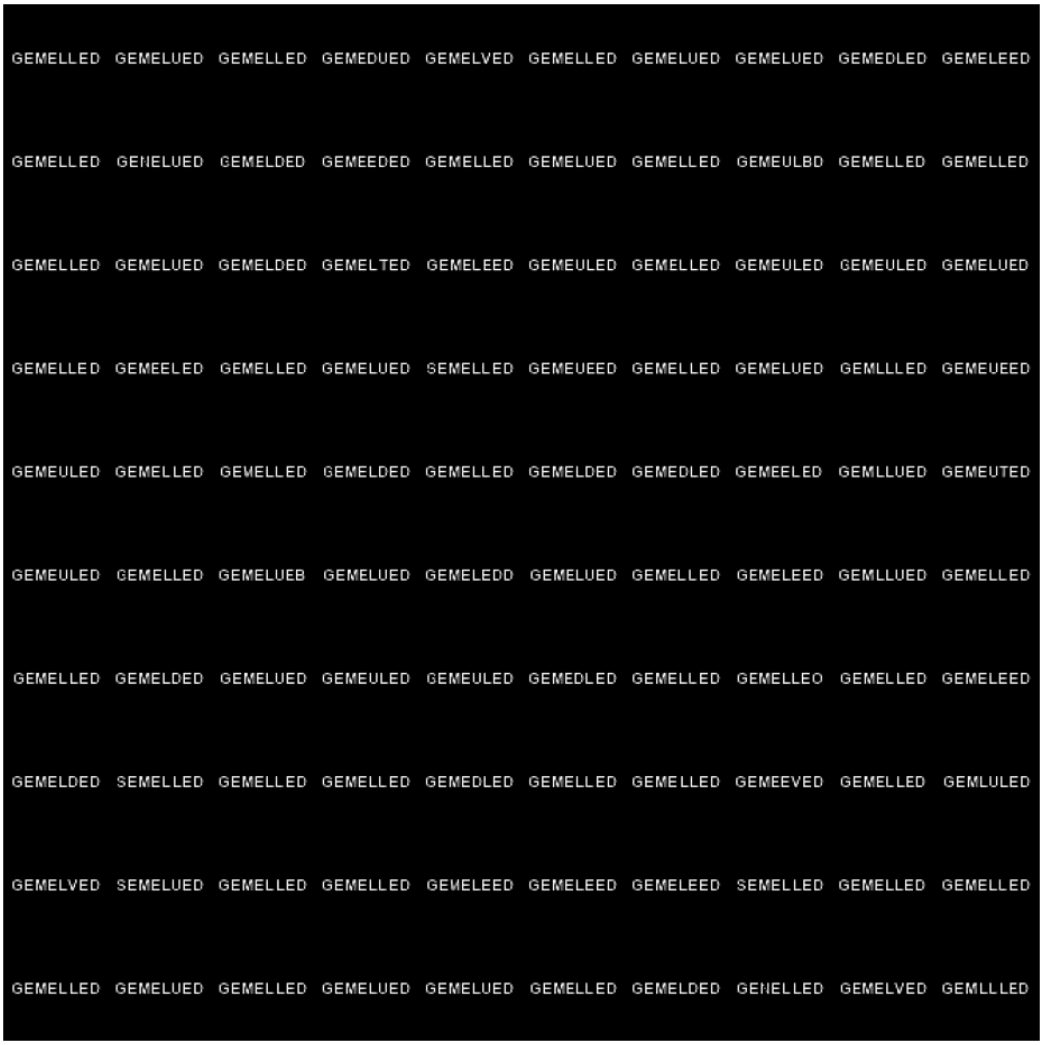
Caption: "a black and white image of the word "GURUNG"
OCR score: 46%



Caption: "a black and white image of the word "SJM"
OCR score: 29%



Caption: "a black and white image of the word "GEMELLED"
OCR score: 4%



Bibliography

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

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.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

OpenAI. Dalle 2 product website. https://openai.com/product/dall-e-2, 2022. Accessed: 2023-05-23.

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.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative adversarial networks. corr abs/1406.2661 (2014). *arXiv preprint arXiv:1406.2661*, 2014.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

DeepFloyd. Hugging face deepfloyd if space. https://huggingface.co/spaces/DeepFloyd/IF, 2023. Accessed: 2023-05-23.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

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.

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

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

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

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

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

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.

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.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022.

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

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.

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

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34:1415–1428, 2021.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi: $10.23915/\mathrm{distill}.00030$. https://distill.pub/2021/multimodal-neurons.

Joanna Materzyńska, Antonio Torralba, and David Bau. Disentangling visual and written concepts in clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16410–16419, 2022.

Giannis Daras and Alexandros G Dimakis. Discovering the hidden vocabulary of dalle-2. *arXiv preprint arXiv:2206.00169*, 2022.

Raphaël Millière. Adversarial attacks on image generation with made-up words. *arXiv preprint arXiv:2208.04135*, 2022.

nostalgebraist. Franks image generation model, explained. https://nostalgebraist.tumblr.com/post/672300992964050944/franks-image-generation-model-explained, Jan 2022.

Rosanne Liu, Dan Garrette, Chitwan Saharia, William Chan, Adam Roberts, Sharan Narang, Irina Blok, RJ Mical, Mohammad Norouzi, and Noah Constant. Character-aware models improve visual text rendering. *arXiv preprint arXiv:2212.10562*, 2022.

Tatu Ylonen. Wiktextract: Wiktionary as machine-readable structured data. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 1317–1325, 2022.

Romain Beaumont. Clip retrieval: Easily compute clip embeddings and build a clip retrieval system with them. https://github.com/rom1504/clip-retrieval, 2022.

Ayush Kaushal and Kyle Mahowald. What do tokens know about their characters and how do they know it? *arXiv preprint arXiv:2206.02608*, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Jeffrey T Morisette and Siamak Khorram. Exact binomial confidence interval for proportions. *Photogrammetric engineering and remote sensing*, 64(4):281–282, 1998.

Alex Clark. Pillow (pil fork) documentation, 2015. URL https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rmi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning

library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Christoph Schuhmann. Laion-400-million open dataset. https://laion.ai/blog/laion-400-open-dataset/, 2021. Accessed: 2023-04-20.

Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*, 2021.

Pamela Mishkin, Lama Ahmad, Miles Brundage, Gretchen Krueger, and Girish Sastry. Dalle 2 preview - risks and limitations. 2022. URL https://github.com/openai/dalle-2-preview/blob/main/system-card.md.

A. Mishra, K. Alahari, and C. V. Jawahar. Scene text recognition using higher order language priors. In *BMVC*, 2012.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022.

AFDELING
Straat nr bus 0000
3000 LEUVEN, BELGIË
tel. + 32 16 00 00 00
fax + 32 16 00 00 00
www.kuleuven.be