

PosterAI: A Unified Image-to-Image Model for High Quality Text-Integrated Image Generation for Posters and Book Covers

Ciana Tzuo

Horace Mann School, 231 W 246th St, Bronx, NY 10471, USA; ciana_tzuo@horacemann.org

INTRODUCTION

Creating images with embedded text, such as posters, flyers, and book covers, often demands significant time, skill, and financial investment. Most individuals lack the expertise of graphic designers or artists, and hiring professionals may exceed the budgets of small business owners and aspiring creators. This limitation hinders their ability to market products effectively or convey their ideas visually. The need for automated tools that can generate images with accurately placed and visually coherent text is growing, particularly in publishing and marketing industries, where text-integrated visuals are crucial for communication.

Self-publishing authors, in particular, face challenges designing book covers that reflect their book's plot while attracting readers. The cover must balance visual appeal and textual information, such as the title and author's name, in a harmonious layout. Reducing the time and cost involved in this design process can empower authors to produce high-quality covers more efficiently and creatively.

Recent advancements in AI image generation, such as DALL-E 2, Imagen, and Stable Diffusion, have significantly improved the quality and realism of synthesized images. However, these tools fall short of the needs of this community in several areas. First, these models struggle with embedding text into images with proper placement, legibility, and coherence. Early research relied on Generative Adversarial Networks (GANs), which faced issues of stability during training. Diffusion models, particularly Latent Diffusion Models (LDMs), have shown promise in generating high-resolution images efficiently. However, integrating text seamlessly into these images remains a challenge.



Figure 1: Limitations with existing GenAI models with embedded text in images; images generated using the prompt: “create a book cover for the invisible man”
AI generated images created by Canva AI, Leonardo AI, and Civitai AI

Second, these models struggle with accurately representing the text, including in multiple languages, as the examples in Figure 1 illustrate.

Finally, these models give limited ability for the creator to control the style of the generated image to match the tone of the content, for example to communicate a business book or a production of a musical from the 50's.

This paper introduces PosterAI, a new innovation that attempts to create a new tool for this community that addresses these challenges. The central research question posed by this paper is: can existing technologies be leveraged to enable users to quickly and efficiently generate high-quality images with accurately embedded and contextually appropriate text in complex visual environments, using both text prompts and image inputs, while maintaining legibility, precise text placement, visual coherence and style control?

To do this, Poster API attempts to combine several existing open source technologies including a general language model (GLM) for interpreting source images; transformer-based layout generation mechanisms for predicting optimal text placement, ensuring coherence with the overall visual design; latent diffusion models (LDM) for generating new images; a character-aware loss function to maintain text clarity and sharpness. Specialized denoising techniques were used to prevent text distortion, ensuring high-quality outputs. Finally, a new custom training set made up of 250 book covers were created using manual data captioning and OCR-based text segmentation.

By combining these technologies in innovative new ways, PosterAI addresses key challenges like text prompt generation, layout control, legibility, and denoising. Users are able to use an image similar to their desired style as an input prompt.

By building on diffusion models and integrating novel methods for text positioning, quality control, and image prompting, PosterAI aims to bridge the gap between image synthesis and text generation, offering a robust solution for automated design tasks.

RELATED WORK

General Image Generation

Recent advancements in text-to-image generation and diffusion models have significantly enhanced the ability to produce high-quality, photorealistic images from textual descriptions. Models like DALL-E 2 (Ramesh et al., 2022), Imagen (Saharia et al., 2022), and Stable Diffusion (Rombach et al., 2022) have demonstrated the power of diffusion-based approaches to create complex visuals guided by text prompts.^{1, 2, 3} These models employ large pre-trained language models and CLIP embeddings to generate detailed images, but they face challenges in integrating text seamlessly within visual contexts, particularly regarding layout control and text coherence.

Diffusion Models/Latent Diffusion Models

Diffusion Models have become a core technology in image synthesis. They are generative models that create images by learning to reverse a gradual noising process. During training, random

Gaussian noise is added until the image becomes a pure noise. The model is then trained to undo the noise step-by-step through a denoising process, refining the image back to its original form. This stepwise approach allows diffusion models to generate images with intricate details and realistic features. The model's ability to produce coherent outputs make them effective for tasks like image synthesis, inpainting, and text-to-image generation.

The introduction of Denoising Diffusion Probabilistic Models (DDPMs) by Ho et al. (2020) laid the groundwork for these techniques, which progressively refine noisy data through iterative denoising.⁴ Further advancements by Nichol and Dhariwal (2021) improved the efficiency and quality of image synthesis, while Latent Diffusion Models (LDMs) (Rombach et al., 2022) optimized the process by operating in lower-dimensional latent spaces, reducing computational overhead while enabling high-resolution image generation.^{5, 3} LDMs operate in a lower-dimensional latent space to improve computational efficiency. LDMs are an extension of traditional diffusion models that operate in a compressed latent space instead of the full-resolution pixel space. During training, an image is first encoded into a lower-dimensional latent representation using a Variational Autoencoder (VAE). The diffusion process then occurs within this latent space, making the generation process much more computationally efficient. Once the denoising process is complete, the latent representation is decoded back into a high-resolution image. This approach allows LDMs to maintain high image quality while reducing the computational cost of generating detailed images. Despite these improvements, embedding legible and well-placed text within images remains a significant challenge.

In the realm of text-to-image generation, models like DALL-E 2 and Imagen excel at generating visuals from prompts but struggle with layout coherence and accurate text rendering. Although these models can produce images containing text, the placement, alignment, and sharpness of the text are often inconsistent. This limitation hinders their application for tasks requiring precise text integration, such as book covers and advertisements.

Transformer-Based Models

Transformer-based models have revolutionized the field of artificial intelligence, particularly in natural language processing (NLP) and beyond. These neural network architectures, introduced in 2017, have become fundamental in various AI applications due to their ability to process and understand sequential data effectively. A notable example is BERT (Bidirectional Encoder Representations from Transformers), which uses deep bidirectional representations to carry out a wide range of tasks.

Layout and OCR Generation

Layout and OCR Generation play a critical role in text-embedded image synthesis. OCR is a technology that converts various types of text-containing images into machine-readable text. Tools like EasyOCR (Jaided AI, 2020) and models like LayoutLM (Xu et al., 2022) have advanced OCR capabilities and layout understanding in structured documents.^{6, 7} Recent research on OCR methods have focused on using segmentation masks and bounding boxes to determine regions of interest (Zhu et al., 2022).⁸

However, these tools primarily focus on recognizing and interpreting existing text rather than generating new layouts. Integrating OCR-based segmentation and layout prediction into the generative process remains an area with untapped potential. This paper focused on bounding boxes and character-level segmentation masks methods to generate layout. Bounding boxes are rectangular markers used in computer vision to define the position and size of objects within an image. They help models identify where specific objects or regions of interest are located, facilitating tasks like object detection, segmentation, and text placement. In text-to-image generation, bounding boxes ensure that text is positioned accurately within an image, preventing overlap with other elements. Accurate bounding boxes are crucial for maintaining legibility and contextual relevance in complex visual designs. Character-Level Segmentation Masks are binary masks indicating where text should appear in the image. Unlike word-level or bounding-box methods, these masks provide fine-grained detail by showing the border of individual character shapes and positions. This approach ensures accurate text placement, legibility, and seamless integration of text with surrounding visual elements. In text-to-image generation, character-level segmentation masks help models avoid issues like overlapping or misaligned characters. They are essential for applications requiring high precision, such as generating complicated typography.

General Language Models

The GLM-4V-9B (Zeng et al., 2024) is an open-source, state-of-the-art multimodal model developed by Tsinghua University, capable of processing both text and visual inputs.⁹ It supports high-resolution images and excels in tasks like image captioning and optical character recognition. This model generates descriptive prompts based on visual inputs, which aids in creating accurate and contextually relevant text layouts. Its bilingual capabilities in English and Chinese enhance its versatility for diverse applications.

METHODS

Despite recent advancements, current AI tools lack the ability to seamlessly integrate text within generated images while maintaining legibility and coherence. Issues like style control, blurred text, inconsistent alignment, and poor layout control remain prevalent. This research paper attempts to address these gaps by creating a new data set of book covers that can be used to train an AI model. It then incorporates layout generation, OCR-based segmentation, and text-aware denoising directly into the diffusion process for creating new images. This approach ensures accurate text placement and legibility, advancing the capabilities of text-to-image synthesis models.

Training Data

The data collection process for PosterAI involved several steps to create a high-quality dataset that would allow the model to accurately learn to generate text in image, with a particular focus on book covers. This section outlines how the data was curated, organized, and processed to

achieve a diverse and contextually rich dataset, supporting both visual and textual learning for the model.

For this project, I started with an open-source image generation model called TextDiffuser from JingyeChen. The open-source model made by JingyeChen used MARIO-10M to train their data.¹⁰ This is a large-scale dataset that includes 10M images with text, including premade OCR for each image that includes character-level segmentation annotations for training. The goal was to enhance this model with a new data set relevant to the project’s specific goals.

Image Collection and Organization

The first step was to select a diverse set of book covers. The dataset used in this research was compiled from publicly available image sources and conforms to the licensing terms provided by the original creators. Images were used solely for research purposes and were modified to include bounding boxes and segmentation masks as part of this study. All images appearing in this paper have been clearly labeled with detailed copyright information.

The diversity of the selection was important to ensure that the model could generalize well across different styles and learn different layouts. This includes covers with varying typography, colors, and text placement. This also included books of different genres and authors. The images were first downloaded as a jpeg file before being processed. Once the images were collected, they were renamed for consistency and placed into organized folders. Each folder corresponded to a specific image and contained additional components required for training, such as captions, OCR (Optical Character Recognition) segmentations, and visual representation of text bounding boxes. The caption provides semantic information to the model and acts as input prompts that guide the mode in generating layout and content. The OCR file helps the model understand where text appears in the image. It stores both the content and the spatial location of the text in the image. This allows the model to learn the placement of text in relation to the visual elements of the image. The character-level segmentation file contains a segmentation mask as a binary image where each pixel either belongs to text or the background. This file helps the model guide the model on where each character should be placed within the image and helps the model focus on the regions of the image that contain text. A NumPy file is lightweight and optimized for storing array data, making it ideal for saving binary masks. This



Figure 2: Collection of some book covers in the training set
Left to Right, Top to Bottom: (Hamid, 2022), (Mlodinow, 2013), (Canetti, 2022), (Bradbury and Steadman, 2003), (Heti, 2022), (Winter, 2022), (Hughey, 2022), (McEwan, 2023), (Thomson and Austen, 1894), (Tolkien, 1995), (Cărtărescu and Carter, 2022), (Thanhauser, 2023)

organization helped maintain consistency and ensured the model could access relevant information during training.

Captions creation

To help the model understand the context and unique features of each book cover, captions were manually created for each image. These captions included information about the plot, which provided background context, unique aspects of the book cover, and all words present on the cover to help the model associate text with visual elements. The captions served as a bridge between the semantic meaning of the text and the visual characteristics of the image. By embedding this information, the model could learn to associate textual elements with visual cues, improving its ability to generate text that complements the image's visual design.

Identifying Text

The next step was to use OCR tools to detect and recognize words present in the image. OCR is a technology that converts various types of text-containing images into machine-readable text. OCR was critical for teaching the model where text appeared relative to the image's other elements. OCR segmentations were created to define the exact location of words. This involved binarization of the input image, deskewing any tilted text, eliminating distortion for clearer text, and adjusting the image to make the text stand out. There was then feature extraction and character recognition, where OCR extracts important features from each segmented character image and compares extracted features with pre-defined templates of characters.

For this task, I used PaddleOCR, an open-source, end-to-end OCR system, for its ability to clearly identify handwritten words or words with hard fonts. This is especially useful for the needs of processing and creating book covers.

PaddleOCR has two main steps: Text Detection and Text Recognition.

The Text Detection stage detects text regions in the input image. First the input image is resized and normalized to bring the pixel values into a suitable range. The image then underwent feature extraction, where a convolutional neural network (RESNET) was used to extract hierarchical features from the image, creating a backbone. These features captured the characteristics of the text and background to make the text detection more accurate. Next, the features from the different layers of the backbone are combined in a multi-scale feature aggregation using a Feature Pyramid Network (FPN). This method ensures that the model detects text with various sizes, such as large headings and smaller text. The model then predicts the regions of the text by creating a probability map that indicates the likelihood of each pixel belonging to a text region. The probability map is then converted into a binary map using Differentiable Binarization (DB) to segment text regions from the background. DB is used in the model as it is suitable for training deep neural networks. It uses a sigmoid function to approximate the binarization process, making it differentiable and trainable end-to-end.

$$P_b(x, y) = \frac{1}{1 + e^{-\beta(P(x, y) - t)}}$$

The result is a pseudo-binary map during training, and a hard binary map during inference. The binary map is then processed to extract contours or polygons that outline the text regions. Contour detection helps identify the boundaries of connected components in the binary map. Polygon approximation helps approximate the shape of text regions using few points. These polygons or bounding boxes are the final output of the detection step. The output is a list of coordinates that correspond to detected text regions, with a confidence score indicating the model's certainty.

The second stage is Text Recognition. This is where PaddleOCR is responsible for converting cropped text regions (from the text detection section) into readable and meaningful text. The model handles variable text lengths and accurately recognizes the text in challenging scenarios.

The outputs of cropped regions of interest containing the text from the text detection module are used as inputs into the text recognition step. Each cropped image is resized to a fixed height while maintaining its aspect ratio. This ensures consistent input dimensions for the recognition model.

The model does a feature extraction (also called the backbone) where important visual features are extracted from the input image. A ResNet (Residual Network) is used during this process where the input image is passed through multiple convolutional layers where these layers capture low-level (e.g., edges) and high-level (e.g., text structures) features. A ResNet (Residual Network) is a deep learning architecture designed to address the challenges of training a very deep neural network. It prevents problems like vanishing or exploding gradients and degradation problems by introducing residual learning, where the network learns to model the residual (difference) between the input and output of a layer, rather than the complete transformation. The cropped image is passed through a convolutional network to produce a 2D feature map. The feature map retains spatial information about the text but compresses the image into a compact, high-dimensional representation.

The “Neck” of the model is sequence modeling, where the model models the sequential nature of text from the 2D feature map. The 2D feature map is collapsed along the vertical axis (height dimension) to create a 1D feature sequence. A BiLSTM (Bidirectional Long Short-Term Memory) layer is applied to capture dependencies between characters in the text sequence. The layer models both forward and backward dependencies in the sequence, improving recognition accuracy for complex text patterns.

The model then goes to the head or prediction step where it maps the 1D sequence of features to a sequence of character probabilities. The model uses a pre-defined character set for the desired language (in this case, English), including alphanumeric characters and special symbols. The output is a probability distribution over all possible characters for each position in the sequence.

Finally, the model takes the raw output of the prediction model, a matrix of probabilities where each row corresponds to a character in the sequence, and uses a Connectionist Temporal Classification (CTC) to decode it. A CTC allows the model to handle sequences of varying length and handles cases where some characters are missing or repeated. The model predicts blank tokens between characters to account for gaps or variations in text length. These blank tokens are removed during post-processing to produce the final character sequence. The recognition

model is trained using CTC Loss that aligns the predicted sequences with the ground truth by summing over all possible alignments.

The final output from this stage is a sequence of characters decoded from the predicted probabilities.

Text Segmentation

OCR segmentations were created to define the exact location of words. This involved binarization of the input image, deskewing any tilted text, eliminating distortion for clearer text, and adjusting the image to make the text stand out. There was then feature extraction and character recognition, where OCR extracts important features from each segmented character image and compares extracted features with pre-defined templates of characters. In the end, bounding boxes around each word or phrase in the image were produced. For each image, the bounding box coordinates of the detected text were saved in a text file (named ocr.txt). This file contained the precise positions of the text within the image, ensuring that the model could later use this information to determine where text should be placed in new image generations. This step laid the foundation for integrating the text layout into the image, teaching the model the relationship between text and its visual environment.

The next task involved creating a NumPy character segmentation file (charseg.npy) using the bounding boxes and coordinates generated during the OCR process. The character segmentation file is a binary mask that highlights the locations of the text within the image.

The first step involves cleaning and preparing the image using binarization to convert the image into black and white and noise reduction to make the text more distinguishable from the background. The process involved isolating the text regions at the character level, converting the image to a black and white gradient. Text areas were marked white, and the rest of the image was black, providing a clear distinction between text and non-text regions.

This segmentation was critical for the training process because it allowed the model to focus specifically on the areas where text should appear. By operating at the character level, the model could achieve fine-grained control over text placement and styling during image generation.

To streamline the training process, an index file (index.txt) was created. This file contained a list of the names of all the images in the dataset. The index file allowed the model to efficiently load and access the correct folders, images, text files, and annotations during training. This ensured a structured approach to data retrieval, minimizing confusion or delays in accessing the correct image-text pairs.

Text Segmentation Method Comparison

Various approaches can be used for text segmentation, each with its advantages and limitations. Several models were compared, and ultimately a hybrid approach was chosen, which combined the strengths of multiple techniques. Here is an overview of the segmentation methods considered.

First, **Bounding Box Segmentation**. This method involves creating rectangular boxes (bounding boxes) around words, phrases, or lines of text. Using a system, like EasyOCR, can generate bounding boxes and coordinates quickly and accurately for simple layouts. However, bounding box segmentation struggles with complex or stylized text layouts and lacks the precision needed for detailed character-level adjustments.

Secondly, **Character-Level Segmentation**. This method isolates each individual character within a region providing a pixel-level precision.¹¹ A binary mask is applied to identify the text regions in the processed image. It offers high precision and a simplistic way to handle complex layouts, but it is computationally more expensive and difficult to implement compared to simpler methods like bounding boxes.

Thirdly, **Region-based segmentation**. This method identifies connected regions in the image that contain text, segmenting these areas based on pixel connectivity and image properties. Region-based segmentation is flexible and offers an adaptability to different text layouts, especially text on complex backgrounds, but it can struggle with intricate text designs and requires manual tuning for each new image type.

Fourthly, **Semantic Segmentation** for text. This method uses a pixel wise classification where each pixel is classified to belong to a certain class, either text or background. The entire image is passed through a CNN or U-Net and is split into two groups. Using models like U-Net, semantic segmentation offers pixel-level accuracy and is highly effective for handling complex text layouts. However, it requires a large amount of training data and comes with a high computational cost, making it less efficient for certain applications.

Lastly, **Hybrid Approaches**. To address the limitations of individual segmentation methods, hybrid approaches combine the benefits from multiple methods.¹²

In the end, the hybrid approach to text segmentation was chosen, combining the advantages of bounding box segmentation and character-level segmentation. The bounding boxes provided an efficient way to roughly locate text within the image, while the character-level segmentation refined these regions for precise text placement. Although this approach requires more computational resources than using bounding boxes alone, it offered a significant improvement in the quality of the text integration, ensuring that the generated images would be visually cohesive and professionally styled.

By combining these approaches, the created text segmentation function is capable of learning to generate text in the correct locations with high precision, making it especially suitable for tasks like book cover design, posters, and advertisements, where text plays a central role in the overall design.

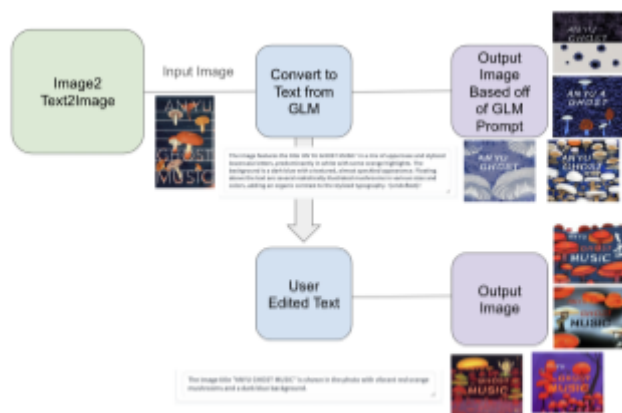


Figure 3: Diagram of the pipeline of PosterAI using the book cover of An Yu's book Ghost Music (2022)

Enabling Image Based Prompts

The creation of the training set turned out to be a manually intensive process that requires a significant amount of time and resources. Results showed that PosterAI's capabilities were very rudimentary when trained with a dataset of 250 book covers.

To overcome these limitations before PosterAI can be properly trained, PosterAI needs to be enhanced to accept an image to use as a starting prompt.

Because the version of the open-source TextDiffuser library that PosterAI is based on can only accept text prompts, PosterAI feeds the image into a GLM which helps produce a detailed text prompt that describes features of the image and key words. The text prompt is then inputted into the training model to create new images.

In this model, the GLM is used to analyze the image for specific characteristics like font and color, and identify the key words, such as titles, in the uploaded image. The GLM then produces a text prompt with the key features and feeds it into the training model.

GLM-4V-9B

After evaluating several GLM's, this project selected GLM-4V-9B. GLM-4V-9B is an advanced open-source multimodal language model from the GLM-4 series, developed by Zhipu AI. It excels in both English and Chinese dialogues, and is able to understand high-resolution images up to 1120×1120 pixels including the ability to effectively handle tasks such as text recognition, chart interpretation, and complex reasoning.

GLM-4V-9B performs image processing by feeding through a visual encoder based on convolutional neural networks like ResNet or transformer-based architectures like Vision Transformers (ViT)). The encoder extracts visual features from the image, turning it into a numerical representation (embedding) that the model can process.

The image is then divided into patches or regions, and each patch is represented as a token using ViT-style patch embeddings. GLM-4V-9B can then perform tasks such as generating descriptive text for an image, or identify text or objects within images

Layout Generation

The first part of the image generation process is layout generation, which focuses on generating the layout of the image, specifically determining the position, size, and orientation of the text within



Figure 4: Diagram illustrating the pipeline for the layout generation using transformer models

the image. This stage is crucial for ensuring that the text is placed in visually coherent locations, aligning with the image's content and structure. This is called a character-level segmentation and is displayed as a box with the word(s) inside the box. Character-level segmentation is used to define bounding boxes around words, which helps the model understand where the text should be located. The segmentations are determined by the following processes: keyword embedding and tokenization and bounding box prediction.

Keyword Embedding and Tokenization

The layout generation starts with the input text prompt, which is processed to identify important words, called keywords. These keywords will determine the position and size of the text in the final image. The first step is tokenization of the input text prompt. Tokenization is the process of splitting a text string into smaller units, called tokens. This is important for splitting the text into manageable pieces, such as words, phrases, or characters, which can be fed into the model, like a Transformer later on in the process, to understand the structure and meaning of the text. In PosterAI, tokenization is performed using a pre-trained tokenizer that splits the input text into tokens that can be processed by the model.

Next, PosterAI feeds the tokenized outputs to be processed in keyword embedding. Keyword embedding is the process of transforming keywords from the input text into dense vector representation. These vectors represent the meaning and context of the keywords, allowing the model to understand which words are more important and how they should influence the layout of the text within the image. Keyword embeddings are generated using an embedding layer in a high-dimensional space, which maps the tokens to vectors that capture both the meaning of the word and their relationship to other words. For example, the words "Mother" and "Parent" may have a strong relationship as they are conceptually related.

Overall, after tokenization, the model identifies keywords. Each keyword is then converted into an embedding (the vectors). The embeddings are generated by passing the tokenized text through an embedding layer. The embedding layer maps each token to a vector that encodes the word's meaning, context, and importance. This is important to PosterAI as it ensures the model can determine which words may need to be bigger as they hold more importance or the placement of the words.

Bounding Box Prediction

After keyword embedding, the next step is predicting the bounding boxes for the position, size, and alignment of the text (keywords) in the layout generation stage. A bounding box is a rectangular frame that binds the objects, in this case the text, in an image. In PosterAI's model, the bounding boxes define the position (location of the text in the image, typically representing the top left corner of the box using (x,y) coordinates), and size (the width and height of the box). For each keyword, the model predicts a bounding box that specifies where the text should go and how large it is.



Figure 5: Bounding boxes shown on a sample image of *Catch-22* (Heller, 1961)

PosterAI uses a Transformer-based model to predict these bounding boxes. The tokenized text and keyword embeddings are fed into a Transformer encoder, which processes the text while taking into account both the semantic meaning and the positional information of each word.

This process happens after tokenization and embedding and predicts the spatial layout using a Transformer-based model. The tokenized text and keyword embeddings are taken from the previous step and passed through the Transformer model.

A Transformer-Based model includes an Encoder and a Decoder. The encoder takes the tokenized text as input and processes it using self-attention mechanisms to capture dependencies between words. The encoder takes input tokens and processes it by taking account both the semantic meaning of the text and the positional information of each word in the high dimensional space.

Because Transformer models do not inherently understand the order of tokens in a sequence, positional encodings are added to the token embeddings. These encodings provide information about the position of each token in the sequence to help the model understand word order and context. This is important as the layout generation needs to know the order of words to generate a meaningful layout.

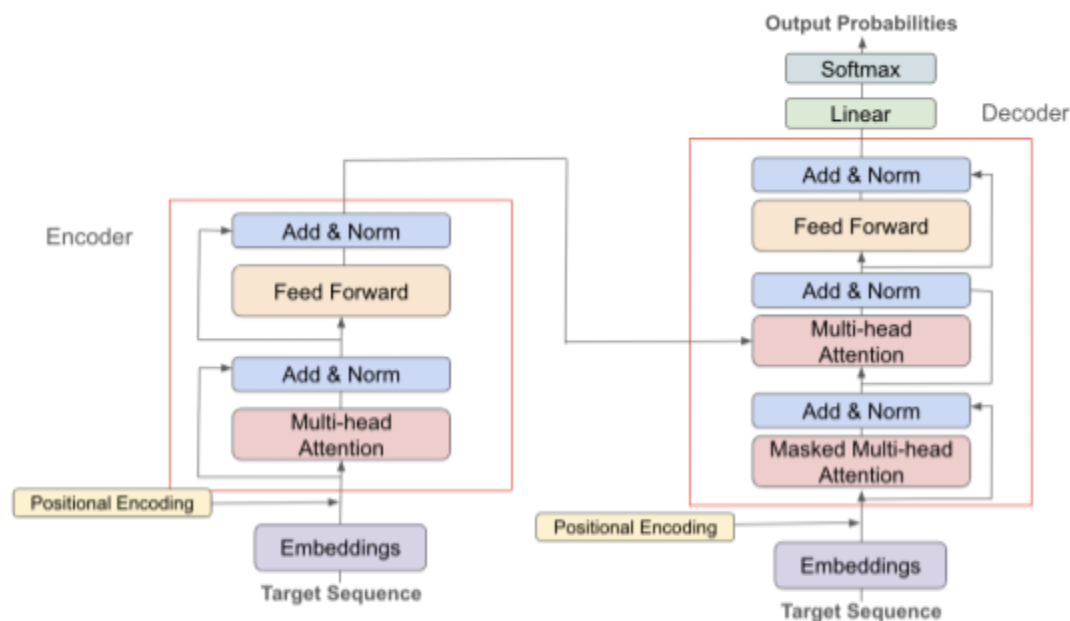


Figure 6: Diagram shows structure of a transformer, including an encoder and decoder

In addition, to help the Transformer focus on different parts of the input sequence based on their relevance to each token, there is a self-attention mechanism. For each token, the model computes an attention score that determines how much focus it should give to the tokens in the sequence. This helps the model understand the dependencies between words, which is important for understanding the over structure of a sentence.

The attention score is computed with Query (Q), Key (K), and Value (V) vectors that are derived from the token embeddings.

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

Where d_k is the dimensionality of the key vectors and the softmax function ensures the attention weights sum to 1.

After the attention step, each token's representation is passed through a feed-forward network (FFN) to further refine the information. The FFN is applied independently to each token, but shares the same parameters across all tokens. The step allows the model to transform the token embeddings to high-level features that can be used by the decoder for generating bounding boxes.

Lastly, to stabilize training and allow the model to learn deep representations, the residual connections prevent the vanishing gradient problem by allowing the model to pass information directly across layers.

The result of the encoder is a sequence of dense vectors (one for each token) which capture both the meaning of the token and their relationship with each other.

PosterAI now feeds the output of the Transformer model encoder to be passed through a decoder, which predicts bounding boxes for the text. The decoder generates one output at a time, with each new output depending on a previous one. This is called autoregressive prediction, where each new box depends on previous boxes and the overall structure of the layout. This helps ensure that the boxes are in logical areas and are visually appealing.

The input to the decoder is the encoded representation from the encoder and partial output sequences (if there are previous predicted bounding boxes). The decoder uses masked self-attention to focus on different parts of the output sequence but also prevents the model from looking at future tokens or outputs during training. This is important as each new output is dependent on the previous outputs.

In addition to self-attention, there are also cross-attention layers, also known as encoder-decoder attention. These layers help the decoder attend to the encoded representations from the encoder and focus on relevant parts of the input sequence to generate meaningful outputs (bounding boxes).

After the tokenized text goes through the encoder-decoder transformation architecture, the final output is the bounding box prediction. For each step in the decoder, it outputs the parameters of the bounding box:

(x, y) are the coordinates of the top-left corner of the bounding box

(w, h) are the width and height of the bounding box

These coordinates are predicted based on the keyword embeddings and the context learned by the Transformer. The model optimizes the positions and sizes to ensure that the text fits within the image.

To train an accurate model, a loss function is applied that compares the predicted bounding boxes with the ground truth. PosterAI's model uses L1 loss (mean squared error) to minimize the difference between predicted and truth bounding boxes. The loss penalizes the large deviations from the correct position and size, guiding the model to be more accurate during training. Loss is important to ensure that the model learns through multiple iterations and creates sensible output.

Looking Ahead: Role of Layout Generation in Image Generation

Once the layout is generated, it is passed as input to the image generation stage, where it is used to create the final image.

Most importantly, it is used to create the Character-Level Segmentation Mask. The predicted bounding boxes are used to create character-level segmentation masks, which tell the model exactly where each letter or character of the text should be placed in the image. The mask is then used by the diffusion model to ensure that the text is rendered in the correct position and is coherent with the rest of the image.

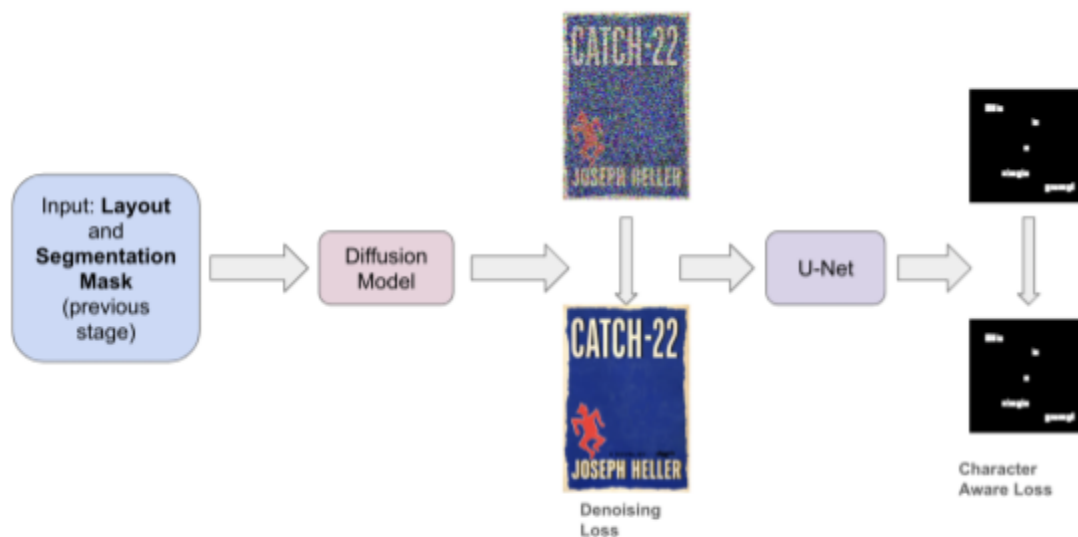


Figure 7: Diagram of the pipeline of the image generation step using the book cover of Heller's *Catch-22* (1961)

Image Generation

After the layout has been generated, the next step is image generation, where the model synthesizes the actual image based on the layouts and segmentation masks generated in the previous stage. This step involves a diffusion-based model that integrates the previous bounding box predictions, character-level segmentation masks, and other layout information to create a

visually coherent image. The model progressively generates the image by removing noise from a latent representation of the image. The main processes during this stage are the latent diffusion model, creating masked features, and training on a character-aware loss to help generate the image with text. The goal is to use the bounding box layout to produce an image that includes text that fits seamlessly into the layout, with the text being legible and visually coherent.

Latent Diffusion Model (LDM)

At the core of the image generation process is the Latent Diffusion Model (LDM). It operates on the latent space of an image, which is the compressed stage of the image produced by the encoder or Variational Autoencoder (VAE).

By operating in the latent space, the model can work more efficiently, as the data is represented in a more compact, lower-dimensional form. The latent features are multiplied by a scale factor to ensure that they are appropriately adjusted before being used in the diffusion model. The LDM adds Gaussian noise to the latent representation and learns to remove the noise step by step to recreate the image with new components.

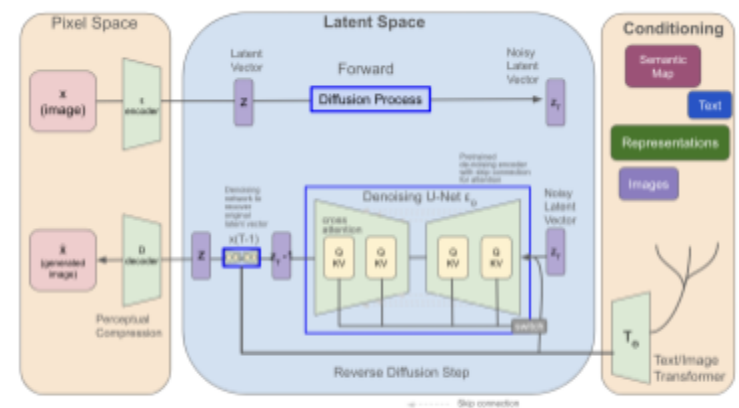


Figure 8: Structure of a latent diffusion model

At each timestep, the model introduces noise into the latent representation using a noise scheduler. The noise scheduler controls how much noise is added at each step. The final goal of the model is to progressively denoise the noisy representation until it becomes a clean image with the added component(s).

For example, during the first timestep, the model might add a significant amount of noise to the latent representation, making the image appear blurry and indistinct. As the model progresses through the timesteps, it removes the noise step by step, gradually refining the image until it becomes clear and fully formed.

Masked Features

When generating an image, it is essential for the model to focus on specific parts of the image, particularly the regions where text should appear. Masked features and feature masks can help guide the model to focus on certain regions.

PosterAI generates masked images by multiplying the original image by a mask that indicates which regions should be hidden. These masked regions are filled in by the model during the image generation process. For example, if the text is supposed to appear in the top-right corner of the image, the rest of the image might be masked during the early stages of generation to allow the model to focus on generating the text first.

Additionally, feature masks are used to indicate which parts of the latent representation correspond to the text. These masks are resized to match the resolution of the latent features and are applied during the denoising process to ensure that the text is placed correctly within the image.

Character-Level Segmentation

The character-level segmentation mask generated in the layout generation stage plays a crucial role in the image generation process. The mask delineates the exact regions where the text should appear, ensuring that the text is placed in accordance with the bounding boxes predicted earlier.

The segmentation mask is derived from the bounding boxes and indicates the regions where text should appear in the final image. Each region corresponding to a character in the text is marked, while the background is left unmarked. During training, the segmentation mask is augmented by randomly applying transformations such as dilations or erosions, which help the model generalize better when dealing with different styles and placements.

The segmentation mask is used to focus the model's attention on text regions during the denoising process. By using the segmentation masks during noise prediction, PosterAI ensures that the text is generated where it should be and is aligned with the layout generation earlier.

Denoising Process

The denoising process is the most important part of the latent diffusion model. It involves progressively refining the noisy latent representation of the image until it reaches a clean, fully formed image.

At each timestep, the model predicts the noise added to the latent features. This prediction is made using a U-Net model, which processes the noisy latents, the current timestep, the encoder hidden states that contain information about the text, and the text embeddings that were generated earlier during layout generation.

A U-Net model is a type of convolutional neural network (CNN) that is well-suited for pixel-wise predictions. It is well-suited for tasks like denoising because it can capture both local and global features of the image, ensuring that the text and the background are generated in harmony. It consists of an encoder and a decoder.

The inputs of the U-Net model include the noisy latent features, which are the latent representation of the image that has been corrupted by noise; timesteps, a value that indicates the current step in the diffusion process; and a text encoder output (the text embeddings), which is taken from the text encoder, and contains semantic information of the input text that can help

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

MSE = mean squared error

n = total number of data points

Y_i = observed values (actual output value)

\hat{Y}_i = predicted values (predicted output value)

the U-Net align the image generation process better. The objective of the U-Net is to learn to predict and remove the added noise over several iterations.

The encoder compresses the noisy latent representation into lower-dimensional representation while keeping the important features. The decoder gradually reconstructs the image by upsampling representation back to its original resolution by restoring spatial details and adding predictions.

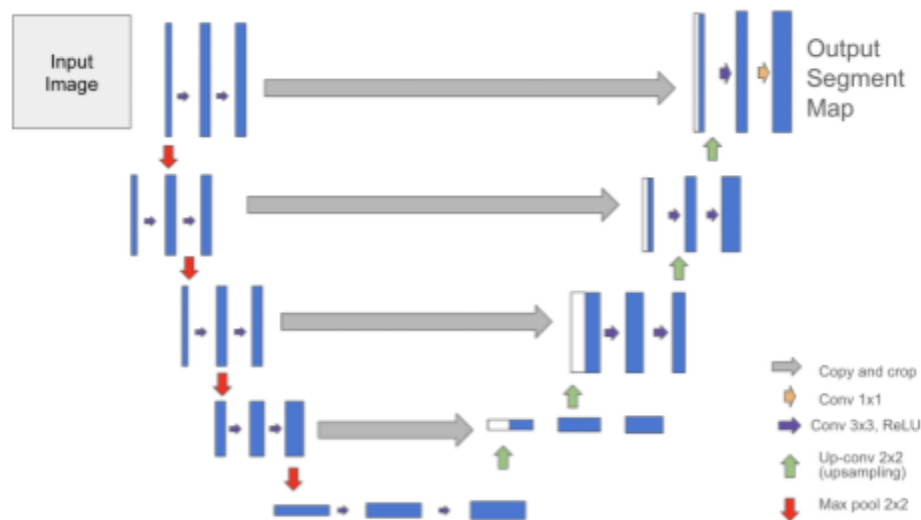


Figure 9: Diagram of an encoder

The U-Net has a convolutional layer in its encoder path that is responsible for extracting features from the noisy latent image representation. The convolutional layers extract important spatial features from the noisy latent image. The U-Net also does downsampling or pooling which is when the feature maps are down-sampled using max pooling operations to catch global features and context.

In the U-Net's decoder path, it gradually up-samples the feature maps back to their original resolution. This process reconstructs the images, removing the noise. At each stage of up-sampling, the feature maps from the corresponding downsampling layer are concatenated with the current up-sampled feature maps. This skip connection helps the U-Net preserve both local details (such as text placement) and global context (overall image structure).

In the end, the U-Net outputs a noisy residual, which is the model's prediction of the noise that was added to the latent image at the current timestep. The noise residual is subtracted from the noisy latent image to produce a cleaner version of the image with less noise. The process is repeated multiple timesteps, with each iteration progressively removing more noise until a clean fully formed image is produced.

The text embeddings generated during layout generation are also fed into the U-Net to ensure that the text is aligned with the rest of the image. This helps the model generate the text in the correct locations and ensures that the text is legible.

Character-Aware Loss

To ensure that the model accurately generates text in the correct regions, a character-aware loss function is introduced. This cross-entropy loss penalizes the model if the predicted text regions do not match the ground truth segmentation mask, helping the model focus on generating high-quality images with text.

By combining the denoising loss or means squared error (which ensures that the image is generated correctly) with the character-aware loss (which ensures that the text is placed correctly), the model learns to remove noise from the latent space and accurately generate high-quality images with text in appropriate locations.

Final Image Output

After the denoising process is complete where the noise has been progressively removed using a U-Net model, the latent representation is still in a compressed form. The final step is to decode the refined latent representation back into pixel space using the VAE decoder.

This process converts the latent features into a high-dimensional image, producing the final output, a clean, fully formed image with accurately placed text according to the bounding boxes and segmentation mask. By using a scaling factor, the decoder ensures that the latent values are converted into appropriate pixel intensities, allowing the image to have the correct brightness, contrast, and color.

For example, if the text is supposed to appear in white on a dark background, the VAE decoder ensures that the final image reflects this design, with the text being legible and visually aligned with the rest of the image.

RESULTS

The project successfully delivered a prototype of PosterAI using the aforementioned technologies.

The prototype wraps the technologies used in a user-friendly workflow.

First, the user uploads an image to use as the inspiration. PosterAI then converts that image into a text prompt.

The user is able to adjust the prompt at that point, for example, editing the text to be used in the generated images.

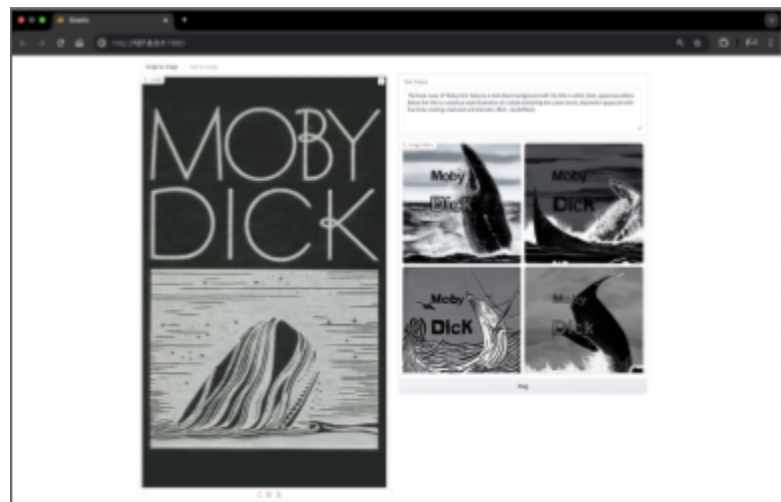


Figure 10: Final result of PosterAI, where a text prompt and images are generated using an input of Melville's Moby Dick (1851)

Finally, four images are produced for the user to select from.

Qualitative Analysis

Qualitative analysis involves visual inspection of generated images to assess their overall coherence and contextual appropriateness. The overall criteria is text placement (Does the text align seamlessly with the image elements?), contextual relevance (Is the text appropriate for the visual content?), and aesthetic quality (Does the text enhance the visual appeal of the image?). The purpose is to provide insights on the output and analyze the quality of the output.




Image	Text Placement	Contextual Relevance	Aesthetic Quality
	<ul style="list-style-type: none">- Good text placement- No words are overlapping	<ul style="list-style-type: none">- The book title and images match	<ul style="list-style-type: none">- The picture is pleasing to the eye
	<ul style="list-style-type: none">- Good text placement- No words are overlapping	<ul style="list-style-type: none">- The book title and images match	<ul style="list-style-type: none">- The picture is pleasing to the eye
	<ul style="list-style-type: none">- Good text placement- No words are overlapping	<ul style="list-style-type: none">- The book title and images match	<ul style="list-style-type: none">- The picture is pleasing to the eye

Figure 11: Table using qualitative analysis to analyze the output images
Images created from PosterAI adapted from Sant-Exupéry's Little Prince (1943),
Mestre-Reed's Sacrificio (2022), and Yu's Ghost Music (2023)

Based on the qualitative analysis, PosterAI is indeed able to achieve the project goals.

Quantitative Analysis

Quantitative analysis involves a more technical inspection of the generated images to assess their overall coherence and contextual appropriateness, especially focusing on the integrated text. A calculation of Intersection over Union (IoU), a standard metric for measuring the accuracy of bounding box placement, will be used to measure the accuracy of the embedded text. IoU is calculated as the ratio of the intersection area between the predicted bounding box and the ground-truth bounding box to the union area of both boxes. This metric evaluates how closely the predicted bounding boxes match the ground-truth placements, ensuring that text appears where it is intended without overlap or misalignment.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

Element	Ground-Truth Box	Predicted Box	IoU
The	(100, 50, 80, 40)	(105, 55, 75, 35)	0.8203125
Little	(200, 50, 120, 40)	(205, 55, 115, 40)	0.7488372093023256
Prince	(150, 100, 150, 50)	(155, 105, 145, 45)	0.87



Figure 12: Example of IoU calculations using Sant-Exupéry’s Little Prince (1943); red box is predicted, green box is ground-truth, and dotted blue is overlapping

The model achieved a mean IoU of 0.82 with a standard deviation of 0.04, indicating consistent performance across all bounding boxes. Overall, most IoU scores were above 0.75, indicating that PosterAI performs well in aligning text with its intended placement. Lower IoU values, such as 0.76 for the word ‘a’, were caused by misaligned bounding boxes due to tighter layouts and more words on the page. To further enhance IoU performance, refining bounding box prediction for smaller or closely spaced text will be prioritized.

CONCLUSION & LIMITATIONS

Assessment of Findings

This paper presented PosterAI, a new innovation enabling anyone to generate high quality images with embedded text for use in Posters, Book Covers, and other similar graphic design creations. The results effectively address the research question on whether existing technologies can be combined to create a new tool that accurately integrates text within images while maintaining legibility, contextual appropriateness and style control. PosterAI leverages diffusion models, GLM models, transformer-based architectures, OCR technologies and introduces character-level segmentation and bounding box prediction to ensure precise text placement and coherence between textual and visual elements.

By building on the strengths of each technology, PosterAI overcomes limitations in previous text-to-image synthesis methods, providing a more reliable and flexible solution.

Limitations and Challenges

Despite the advancements, PosterAI has several limitations and challenges:

- **Limited training data set.** The model should continue to be refined with larger and larger data sets, especially examples of target media such as book covers, movie posters, theater posters, ads, and more.
- **Complex Backgrounds:** The model may struggle with highly detailed or dynamic backgrounds, where maintaining text legibility becomes difficult.
- **Computational Overhead:** The two-stage process of layout generation and image synthesis requires significant computational resources, which may limit real-time or large-scale deployment.
- **Text Style Diversity:** While the model performs well for standard fonts and styles, it may encounter difficulties with extreme variations in handwriting, calligraphy, or decorative text.
- **Bounding Box Accuracy:** For longer or multi-line text, bounding box predictions can sometimes result in misalignment, overlapping, or crowding of elements.

Unforeseen issues, such as the model's sensitivity to variations in input text length or context, also emerged during experimentation. Extending this work to handle a broader range of text styles and more complex visual contexts will require further research and optimization.

Potential Applications

PosterAI has potential applications across several industries and use cases:

- **Graphic Design:** Automating the generation of book covers, posters, and marketing materials where accurate text placement is crucial.
- **Advertising:** Creating customized advertisements with text that integrates seamlessly into product images and visuals.
- **Publishing:** Enhancing layout design for magazines, e-books, and print media by automating text integration within complex visuals.
- **Personalized Content Generation:** Crafting personalized notes, invitations, or digital content that blend text with imagery.

By addressing the complexities of text-to-image synthesis, PosterAI offers a versatile and efficient tool for industries that rely on high-quality visual content. Its ability to generate coherent and contextually appropriate text-integrated images paves the way for more innovative applications in design automation and creative content production.

In summary, PosterAI represents a significant step forward in text-to-image generation, combining advanced techniques to produce images where text is accurately placed and visually appealing. Future enhancements could further expand its capabilities and applicability across diverse domains.

APPENDIX

The photos used to train the AI model were publicly available images obtained from Google using the keyword “book cover images.” The purpose of using these images is solely for learning and not for any commercial purposes. The output photos were generated by my self-trained AI model, "PosterAI." Similarly, the purpose of generating these output images is purely for learning and not for any commercial purposes. All images appearing in my paper and other competition materials have been clearly labeled with detailed copyright information.

REFERENCES

1. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical Text-Conditional Image Generation with CLIP Latents. *Computer Vision and Pattern Recognition*. <https://doi.org/10.48550/arXiv.2204.06125>
2. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., & Norouzia, M. (2022). Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. *Computer Vision and Pattern Recognition*. <https://doi.org/10.48550/arXiv.2205.11487>
3. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. *Computer Vision and Pattern Recognition*. <https://doi.org/10.48550/arXiv.2112.10752>
4. Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. *Machine Learning*. <https://doi.org/10.48550/arXiv.2006.11239>
5. Nichol, A., & Dhariwal, P. (2021). Improved Denoising Diffusion Probabilistic Models. *Machine Learning*. <https://doi.org/10.48550/arXiv.2102.09672>
6. Jaided AI. (2020). *EasyOCR*. Jaided AI. Retrieved December 1, 2024, from <https://www.jaided.ai/easyocr/>
7. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020). LayoutLM: Pre-training of Text and Layout for Document Image Understanding. *Computation and Language*. <https://doi.org/10.48550/arXiv.1912.13318>
8. Zhu, W., Sokhandan, N., Yang, G., Martin, S., & Sathyanarayana, S. (2022). DocBed: A Multi-Stage OCR Solution for Documents with Complex Layouts. *Computer Vision and Pattern Recognition*. <https://doi.org/10.48550/arXiv.2202.01414>
9. Zeng, A., Xu, B., Wang, B., Zhang, C., Yin, D., Zhang, D., Rojas, D., Feng, G., Zhao, H., Lai, H., Yu, H., Wang, H., Sun, J., Zhang, J., Cheng, J., Gui, J., Tang, J., Zhang, J., Sun, J., . . . Wang, Z. (2024). ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools. *Computation and Language*. <https://doi.org/10.48550/arXiv.2406.12793>
10. Chen, J., Huang, Y., Lv, T., Cui, L., Chen, Q., & Wei, F. (2023). TextDiffuser: Diffusion Models as Text Painters. *Computer Vision and Pattern Recognition*. <https://doi.org/10.48550/arXiv.2305.10855>
11. Fateh, A., Fateh, M., & Abolghasemi, V. (2023). Enhancing optical character recognition: Efficient

techniques for document layout analysis and text line detection. *Engineering Reports*, 6(9).
<https://doi.org/10.1002/eng2.12832>

12. Shehzadi, T., Stricker, D., & Afzal, M. Z. (2024, April 30). *A Hybrid Approach for Document Layout Analysis in Document images*. Arxiv. Retrieved December 1, 2024, from <https://arxiv.org/html/2404.17888v2>