TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Major Project Mid-Term Report Part B

On

चित्रण: An Automated Festive Poster Generator with Wishes in Nepali Language

Submitted By:

Kristina Bhandari (THA077BCT022)

Kristina Ghimire (THA077BCT023)

Pradeepti Dongol (THA077BCT033)

Namita Bhatta (THA077BCT048)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

December, 2024

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Major Project Mid-Term Report Part B

On

चित्रण: An Automated Festive Poster Generator with Wishes in Nepali Language

Submitted By:

Kristina Bhandari (THA077BCT022)

Kristina Ghimire (THA077BCT023)

Pradeepti Dongol (THA077BCT033)

Namita Bhatta (THA077BCT048)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Computer Engineering

Under the Supervision of

Er. Saroj Shakya

December, 2024

ACKNOWLEDGEMENT

We extend our heartfelt gratitude to the **Institute of Engineering, Tribhuvan University**, for including our major project in the Bachelor's in Computer Engineering curriculum. This opportunity has significantly enriched our knowledge and skills.

We would sincerely like to thank our supervisor **Er. Saroj Shakya** for his invaluable guidance and feedback on this project.

We express our sincere appreciation to the **Department of Electronics and Computer Engineering, Thapathali Campus**, for their unwavering support and encouragement. Their commitment to providing a quality learning experience throughout our journey has been invaluable.

Furthermore, we would like to thank the research article authors, classmates, teachers, and everyone else who helped us develop our project, directly or indirectly.

Kristina Bhandari (THA077BCT022)

Kristina Ghimire (THA077BCT023)

Pradeepti Dongol (THA077BCT033)

Namita Bhatta (THA077BCT048)

ABSTRACT

The regular design of festival posters can be tedious and time-consuming, particularly for Nepalese people where more than 50 festivals are celebrated. This system aims to automate the creation of Nepalese festival posters in Nepali text by integrating Natural Language Processing with Computer Vision. The system accepts two user prompts in English language one for input to generate Nepali wishes and another prompt for image generation. The system works in two phases: first, it converts an English prompt into a Nepali festival wish using a fine-tuned M2M-100 model. In the second phase, a pre-trained Latent Diffusion Model generates an image based on the user prompt. The generated image's color style is then applied to the text and the text's position is determined through saliency mapping. Finally, the styled text and image are integrated to produce a complete festival poster. This method simplifies the manual process of creating festival posters, enhancing efficiency and creativity. It is a valuable tool for graphic designers and anyone looking to quickly and effectively generate festive posters without prior design knowledge.

Keywords: Computer Vision, Festival poster, Latent Diffusion Model, M2M-100 Model, Natural Language Processing, Nepali Text, Saliency Mapping

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADA	Adaptive Discriminator Augmentation
AdaIN	Adaptive Instance Normalization
AttnGAN	Attention Generative Adversarial Network
AuPoD	Automatic Poster Design
BERT	Bidirectional Encoder Representation from Transformers
BLEU	Bilingual Evaluation Understudy
BP	Brevity Penalty
BPE	Byte Pair Encoding
CGL-GAN	Composition-aware Graphic Layout Generative Adversarial Network
CIFAR	Canadian Institute for Advanced Research
CLAHE	Contrast Limited Adaptive Histogram Equalization
CLIP	Contrastive Language-Image Pre-training
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CSS	Cascading Style Sheet
CSV	Comma-Separated Values
CTR	Click-Through Rate
CUB	Caltech-UCSD Birds
DAMSM	Deep Attention Multimodal Similarity Model
DC-GAN	Deep Convolutional Generative Adversarial Network
DDPM	Denoising Diffusion Probabilistic Model
DF-GAN	Deep Fusion Generative Adversarial Networks
FFHQ	Flickr-Faces-HQ
FFN	Feed Forward Network
FID	Frechet Inception Distance
GANs	Generative Adversarial Networks
GLIDE	Generative Latent Image Disentanglement
GUI	Graphical User Interface
HTML	HyperText Markup Language
ICVT	Image Captioning Visual Transformer
IDE	Integrated Development Environment
IS	Inception Score

KL-Reg	Kullback-Leibler Regularization
LAION	Large-scale Artificial Intelligence Open Network
LDM	Latent Diffusion Model
LLMS	Large Language Model Super-resolution
LSTM	Long Short-Term Memory
LSUN	Large-scale Scene UNderstanding challenge
M2M	Many to Many
MA-GP	Mutual Alignment Gradient Penalty
mBART	Multilingual Bidirectional Auto-Regressive Transformers
MFEN	Multimodality Feature Extraction Net
MLP	Multi-Layer Perceptron
MMCQ	Modified Median Cut Quantization
MMT	Multilingual Machine Translation
mT5	Multilingual Text-to-Text Transfer Transformer
NLP	Natural Language Processing
NLTK	Natural Language ToolKit
OCR	Optical Character Recognition
PIL	Python Imaging Library
PNG	Portable Network Graphics
PSD	Photoshop Document
ReLU	Rectified Linear Unit
RGB	Red Green Blue
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RPM	Revenue per Mile
SAP	Style Attribute Predictor
SOTA	State-of-the-art
TER	Translation Error Rate
t-SNE	t-distributed Stochastic Neighbor Embedding
UCSD	University of California San Diego
VAE	Variational Autoencoders
VGG-16	Visual Geometry Group 16
ViT-S	Vision Transformer Small
VQGAN	Vector Quantized Generative Adversarial Network
VQ-Reg	Vector Quantization Regularization

1. INTRODUCTION

The regular design of festival posters is tedious and time-consuming, particularly for Nepalese people where more than 50 festivals are celebrated. Each festival has its importance and represents the diversity of Nepalese culture. Wishing and celebrating one another festival is a beautiful form of showing respect for each other's culture and tradition. With the increasing popularity of digital platforms, creating festival posters and posting them to celebrate with everyone is trending. However, regularly creating visually appealing festival posters needs some knowledge of design and time. Adding Nepali text makes the process more difficult and time-consuming.

To address this, we are working on a project titled '**चित्रण**: An Automated Festive Poster Generator with Wishes in Nepali Language'. Chitran creates beautiful posters with Nepali texts for Nepali festivals (Dashain, Tihar, Chhath, New Year, and Holi). It takes prompts in English and creates visually appealing posters that include festive wishes in Nepali text, based on the prompts given. With Chitran, celebrating and sharing the joy of Nepali festivals becomes simpler and more visually engaging, while keeping the cultural essence in every poster.

1.1 Background

With the advancement of Natural Language Processing, understanding human intention through text prompts and generating a title is possible. Also, generative models help to create quality images and styles related to the festival. Combining NLP with generative models will provide an effective approach to making the festival posters with less user interaction. Previously, research has been done to create posters captions or images using generative models. Some have created posters in other languages aside from English. However, no research has been done to create Nepali festival posters with Nepali text.

1.2 Motivation

In the era of social media, where social media presence and visual communication immensely impact audience engagement, firms, clubs, and organizations are constantly seeking ways to connect with their audiences. In recent years, one of the most common methods of doing so across multiple domains of Nepali social media accounts has been

through festival posters. However, traditional approaches to poster design present several challenges. Hiring graphics designers for repetitive tasks as such can be costly, tedious, and inefficient along with neglecting resources and time due to its extensive manpower and time investment. Consequently, valuable manpower and funds are diverted to a less prioritized and redundant task which is most likely to harm the performance of the company.

These reasons underscore the need for automation of festival posters, enabling organizations to generate as many festival posters with just a click of a button. Thus, automation of festival social media posters creation can bring about significant improvement in an organization's performance by redirecting the economy and manpower towards more strategic and impactful activities.

1.3 Problem Definition

The festival social media posters generation is a repetitive task. Using the traditional approach for such redundant tasks results not only in improper allocation of resources, manpower, and economy within the organization ultimately resulting in inefficient performance but also underestimates the potential of the hired professional. Hence, the redundant task of designing each festival's social media posters is alleviated by automating the system. The automated process can generate images and text and integrate them to create a cohesive and aesthetically pleasing posters.

1.4 Objectives

- To analyze input prompt to extract event and year details, then generate concise Nepali short title.
- To create Nepali festival-themed image and integrate it with styled titles in Nepali to create a digital poster.

1.5 Scope and Applications

With the help of generative models and natural language processing, this project attempts to create an automated system that, for small enterprises, graphic artists, and digital creators, can produce visually appealing festival postings in Nepali text.

1.5.1 Scope

Automated Festive Poster Design:

The system will automatically generate aesthetically pleasing poster designs for Nepali festivals (Dashain, Tihar, Chhath, New Year, and Holi) based on text prompts provided in English. It will translate the input text into Nepali script and incorporate it into the poster along with relevant images.

Text-to-Image Synthesis

Producing high-quality images from textual descriptions by applying pre-trained models and adjusting these models so they can comprehend and represent different holiday themes and cultural quirks with accuracy.

Scalability and Efficiency

The system can handle large-scale usage, particularly during peak festive seasons optimizing computational resources to maintain efficiency and responsiveness.

1.5.2 Applications

The Festive Poster Design project has broad applications across various domains.

Personal Use

This system enables users to design their own holiday cards and posters to send via email, publish on social media, or print off in physical cards.

Business and Corporate Use

It enables companies to create personalized holiday posters for staff greetings, marketing initiatives, and promotional materials assisting businesses in maintaining cultural relevance and engagement.

Educational Institutions

Help academic institutions rank and analyze student resumes for co-ops, internships, and post-graduation jobs.

2. LITERATURE REVIEW

Angela Fan et al. [1] depict the Many-to-many MMT model that translates between any pair of languages. It can directly translate from 100 languages to 100 languages or 9900 direction combining dense and sparse language-specific parameters to translate directly between languages. The underlying dataset for M2M-100 was mined from CommonCrawl using a novel strategy that exploits language groupings. It has 7.5B training sentences for 100 languages. The model has the size of 15.4B parameters. Like other state-of-the-art machine translation systems, M2M-100 uses a subword segmentation method and a Transformer-based encoder-decoder architecture. The encoder transforms the source token sequence into a sequence of embeddings of the same length, while the decoder produces the sequence of target tokens. Similarly, the input and output of translation systems are sequences of tokens. These tokens are units from a dictionary built to reconstruct any sentence in any language. Moreover, it integrates a mixture-of-experts approach which allocates different subsets of parameters to specific language groups and enhances the model's ability to handle diverse linguistic features The M2M-100 model outperforms a lot of English-Centric multilingual models by improved BLEU scores. It offers faster and more accurate translations by translating directly between languages without using English as a translator. Its scalable architecture also allows several languages and translation orientations to be supported without increasing the training difficulty.

Jonathan Ho et al. [2] proposed the DDPMs whose methodology involves two primary processes: the forward diffusion process and the reverse denoising process. In the forward process, Gaussian noise is progressively added to the data, transforming it into a nearly Gaussian distribution. This process is mathematically designed to ensure a smooth transition from the original data to noise. The reverse process involves training a model to denoise the data step-by-step, effectively reversing the diffusion process. This denoising training process involves optimizing a simplified variational bound objective, while the sampling process uses a trained model to iteratively remove noise from a Gaussian starting point to generate new data samples. On the CIFAR10 dataset, DDPMs achieve an Inception score of 9.46 and a state-of-the-art Fréchet Inception Distance (FID) score of 3.17, surpassing many existing models including StyleGAN2 + ADA (v1).

Bowen Zhang et al. [3] proposed a transformer-based GAN model named StyleSwin for generating high-resolution images. The generator part of GAN uses a Swin transformer. To generate more accurate images, they proposed double attention which covers contexts of both local and shifted windows. They chose AdaIN for the style injection process. They trained and validated the model on various datasets like CelebA-HQ, LSUN Church, and FFHQ. They evaluated the quality of the image based on the FID score. During the training they used Adam solver with beta1 0.0 and beta2 0,99, They set a 5e-5 learning rate for the generator and 2e-4 for the discriminator. They used standard non-saturating GAN loss with an R1 gradient penalty. They used 8 32GB V100 GPUs to fit 32 images as one training batch for 256 x 256 resolution and only 16 images for 1024 x 1024 resolution. They augmented their data as transformers need large datasets which may lead to overfitting. This model with less computational resources performs better than StyleGAN on the CelebA-HQ 1024 dataset achieving an FID of 4.43 and approaches the performance of StyleGAN2 on FFHQ-1024 achieving an FID of 5.07. This model achieved state-of-the-art quality in generating 256 x 256 images and can also be used for high-resolution images of 1024 X 1024.

Shunan Guo et al. [4] developed a system named Vinci that automatically generated advertising posters using deep generative model Sequence-to Sequence Variational Autoencoder (VAE). The poster was generated based on a product image and many taglines. They also added an online editing feature that allows users to edit the generated poster and provide feedback. This model suggests that it requires relatively less training data compared with GAN based techniques to generate high quality posters. The image features were extracted using VGG-16 pretrained on the ImageNet. This system uses human-designed posters that are stored in a layered format like PSD format to extract design elements. To generate the training samples they labeled 3227 posters generated from Vinci automatically. For training, they collected and generated 173 posters of Chinese seafood and soft drinks. Similar to this system there is the Luban system and they have made comparisons with that system. Further work can be done on this system like diverse datasets and various font face designing and creating more complex systems to generate designs like infographics.

A. Ramesh et al. [5] proposed a two-stage model. Firstly, text captions are encoded into CLIP (Contrastive Language-Image Pre-training) embeddings using a prior model.

Secondly, a decoder generates an image conditioned on image embeddings. A generative model of images given captions, P(x|y) is formed by combining these components. Autoregressive prior and diffusion prior models are explored. Diffusion prior showed better performance and computational efficiency. This approach preserves both semantics and style by making use of robust image representations learned by CLIP. Additionally, 3.5 billion parameter GLIDE model is used as the decoder to produce high-resolution images, and a diffusion upsampler model is used for image upscaling. The models are trained on a combination of CLIP and DALL-E datasets. However, the paper identifies several limitations including attribute binding issues, challenges with details in complex scenes. Moreover, CLIP embedding does not precisely encode spelling information of rendered text.

R. Rombach et al. [6] proposed a model called the Latent Diffusion model. The Latent Diffusion model achieved a new state-of-the-art (SOTA) score for class-conditioned image generation and image inpainting. This also reduces computational complexity, unlike other diffusion models. Generating images using the Latent Diffusion model involves denoising the noise sample multiple times to generate high-resolution images that resemble the training data. The model conditions this process of generating images using input prompts. A text-to-image model with 1.45 billion parameters is trained using language prompts from the LAION-400M dataset. It utilizes a BERT-based tokenizer and a transformer to generate a latent code, which is then integrated into a UNet architecture using cross-attention. The use of cross-attention in the Latent Diffusion model enhances the ability of the model to process the diverse input information effectively for conditiondal image synthesis. The models were trained on an input resolution of 256 x 256 using an A100 GPU but can generalize to larger resolutions like 512 x 512 and 1024 x 1024.

D. Huang et al. [7] proposed a saliency-free poster generation system based on the self-supervised end-to-end neural network by dividing the process into two sub-tasks: layout prediction and attributes identification. The overall objective of this project was to minimize the sum of losses of the two sub-tasks. Over 6000 semi-structured posters were collected for this project out of which 4600 posters were used in 7:1:2 ratio for training, validation, and testing. The dataset contained varying topics like portraits, animals, food, scenes, etc. AdamW optimizer was used to train the entire framework.

The system accepts background image and text as input to generate a poster. Multi-modality feature extraction net (MFEN), an encoder-decoder architecture, was utilized to extract image feature, using a convolution-based deep neural network, and text features, using a bert-base-uncased model, to aggregate them using a decoder to generate a density map of the same size as the original image. The density map was combined with original visual features for attribute identification. The position and size of textual box was predicted by solving an optimization problem on the density map. The model was tested against various saliency-based models, human generated posters and a series of varieties of AuPoD frameworks on test loss, Jaccard similarity, macro-averages accuracy and manual rating where AuPoD proved to be better than saliency-based method but still lacked some stability compared to human generated output. The posters generated by the AuPoD framework had an acceptable rate of 94%, followed symmetry property and implicitly constructed aesthetic constraints.

Ming-Tao et al. [8] present the challenges in generating high-quality images from textual descriptions. Traditional GANs used for text-to-image synthesis led to several issues that introduced entanglements between generators of different image scales, limited supervision due to fixed extra networks, and computationally expensive cross-modal attention restricting its use to specific scales. Meanwhile, the DF-GAN (Deep Fusion Generative Adversarial Networks) model provides advanced text-to-image synthesis, offering a straightforward effective approach. DF-GAN overcomes the limitations of traditional GANs by introducing three key innovations. It includes a One-Stage Backbone, Target Aware Discriminator, and Deep-Text Image Fusion Block. One-Stage backbone synthesizes high-resolution images directly avoiding entanglement issues of multi-stage architecture. Similarly, Target-Aware Discriminator improves semantic consistency without the need of extra networks by using a Matching-Aware Gradient Penalty (MA-GP) and one-way output ensuring smooth loss surface at actual and matched data points. DFBlock enhances the fusion of textual and visual elements, allowing for a more thorough integration of text information at different image sizes. However, DF-GAN produces sentence-level text information restricting detailed visual feature synthesis. This issue can be addressed by introducing pre-trained LLMs.

A. Sauer et al. [9] combined the StyleGAN architectures with CLIP embeddings, for text-to-image synthesis. StyleGAN-T attempts to improve semantic alignment between text descriptions and generated images by embedding text prompts using a pre-trained CLIP text encoder and feeding these embeddings into both the generator and discriminator. This paper highlights the computational drawbacks of translational equivariance which is not necessary in text-to-image synthesis, choosing instead to use the more effective StyleGAN2 backbone to lower computing expenses and make it easier to introduce stochastic fluctuations in low-level features by using output skip connections and spatial noise inputs. It introduces two changes to the generator architecture: an easily optimizable residual block to normalize input and scale the output, and stronger conditioning that improved the FID and CLIP score by ~10%. The discriminator has been redesigned with a ViT-S for the feature network. Furthermore, guiding the text encoder and the generator and explicit truncation further boosted the quality and alignment for a given sample. However, while CLIP-based guidance enhances text alignment, it may cause artifacts in images; hence, it may be necessary to retrain CLIP on data with better resolution to address these problems. In addition, the paper notes persistent issues such attribute binding and text coherence within images and suggests future options for further conditioning mechanism enhancements.

J. Lin et al. [10] proposed an automated poster generation system that takes product images and product descriptions and input to produce posters of varying sizes as the result. The system consisted of four key stages: image cleaning and retargeting, layout generation, tagline generation, and style attribute prediction. Image cleaning involves erasing graphic elements on the product image while image retargeting involves adjusting image size. The methods utilized for the first step were graphic element detection, inpainting, outpainting, saliency detection, cropping, and scaling. The second step, layout generation, utilized transformer-based CGL-GAN (Composition-aware Graphic Layout GAN) and ICVT to generate a content-aware poster layout. The tagline generation step used Content-driven Dense-Captioning on Images, a multi-modal generative model. Style attribute predictor (SAP), a transformer-based model, could simultaneously predict font typography, dominant color, stroke color, and gradient color of taglines. 62 commonly used font typography was utilized for tagline font. The project also contributed a dataset of 76,960 posters annotated for tagline content, graphic element position, style attributes, and font typography. A font-weight-aware font

classifier using a self-supervised approach was trained for font typography. Upon comparison with MMCQ, Retrieval, and Classifier methods, AutoPoster produced better R1, R1+R2 scores along with better D-ab and D-light metrics. Additionally, upon comparison with Luban and posters designed by professionals on a voting basis consisting of 129 participants, 20 professionals, and 109 non-professionals, AutoPoster exhibited higher R1 (78%) and R1+R2 (58%) metrics. Furthermore, AutoPoster was compared with posters generated by professionals on 80,000 products where AutoPoster had a higher CTR (click-through rate) and RPM (revenue per mile) increase of 8.24% and 8.25%.

Table 2-1: Summary Table of Literature Review

Title	**Methodology**	**Inferences**
Beyond English-Centric Multilingual Machine Translation (21 Oct, 2020) **Author: Angela Fan et al.**	The model uses transformer architecture with dense scaling and language specific sparse parameters.	• Allows direct translation between any pair of 100 languages. • Achieves over 10 BLEU points higher than English-centric models for non-English language pairs. • Reduces biases from using English as an intermediary and handles large-scale translations efficiently.
Denoising Diffusion Probabilistic Models (DDPM) (16 Dec, 2020) **Author: Jonathan Ho et al.**	Uses a forward diffusion process adding Gaussian noise to data and a reverse denoising process to generate data by removing the noise.	• Achieves high-quality image generation with stability and diversity, leveraging a probabilistic framework for better mode coverage than GANs. • Computationally intensive; requires large amounts of

		data, GPU resources and time for training.
StyleSwin: Transformer-based GAN for High-resolution Image Generation (21 Jul, 2020) **Author: Bowen Zhang et al.**	The generator model is a Swin Transformer and the Discriminator model is a wavelet discriminator.	• Resolution: 256 x 256 SOTA • To generate more accurate images, they proposed double attention. • Less computational resources than StyleGAN.
Vinci: An Intelligent Graphic Design System for Generating Advertising Posters (2021) **Author: Shunan Guo et al**	The model uses a deep generative model Sequence-to Sequence Variational Autoencoder (VAE).	• Takes an input image and extracts image features using VGG-16 pre-trained on the ImageNet. • Has an online editing feature for feedback. • Requires less training data and computational resources than GANs.
Hierarchical Text-Conditional Image Generation with CLIP Latents (13 April, 2022) **Author: Aditya Ramesh et al.**	Uses a two-stage model combining CLIP embeddings with diffusion models. Also, Experiments with both autoregressive and diffusion models for the prior, found diffusion models more efficient and better in terms of sample quality.	• Achieves high-quality image generation in zero-shot settings on the MS-COCO dataset. • Human evaluators significantly prefer this model's outputs over prior state-of-the-art methods.Computationally intensive and requires significant resources for training.

High-Resolution Image Synthesis with Latent Diffusion Models (13 April, 2022) **Author: Robin Rombach et al.**	The model denoises the noise samples multiple times to generate high-resolution images. Also, the model uses cross-attention layers to allow conditioning input.	• Achieves State of the art for class-conditional image synthesis and image inpainting. • Latent-based diffusion model • Less computational requirements than pixel-based diffusion models.
AUPOD: End-to-End Automatic Poster Design by Self Supervision (28 Apr 2022) **Author: Dongjin Huang et. al.**	The model is a self-supervised layout prediction and attributes identification model trained in semi-structured posters. The image encoder is a convolution-based deep neural network and the text feature encoder is a pre-trained contextual token embedding. It uses multilevel perceptron (MLP) to convert textual representation to vector space	• Density mapping for layout prediction • Uses a unified neural network for layout prediction and attributes identification. • Performs better than the saliency-based method on layout prediction sub-task. • Higher variance is score than manually generated posters
DF-GAN: A Simple and Effective Baseline for Text-to-Image Synthesis(15 Oct,2022) **Author: Ming Tao et al.**	Target-Aware Discriminator with Matching-Aware Gradient Penalty and One-Way Output	• Outperforms state-of-the-art models (e.g., AttnGAN, DM-GAN) in image quality and text-image semantic consistency. • Achieves better IS and FID on CUB and COCO datasets. • Simpler and more efficient with fewer parameters and no extra networks.

StyleGAN-T: Unlocking the Power of GANs for Fast Large-Scale Text-to-Image Synthesis (23rd Jan, 2023) **Author: Axel Sauer et. al**	Generator with residual convolutions and stronger text conditioning, while the discriminator employs a ViT-S feature network and applies augmentation techniques to improve performance. It also employs CLIP-based guidance during training to align generated images closely with textual descriptions.	• Outperforms distilled diffusion models at lower resolution in terms of quality and speed. • Computationally intensive, requiring significant resources for training and inference.
AutoPoster: A Highly Automatic and Content-aware Design System for Advertising Poster Generation (23 Aug 2023) **Author: Jinpeng Lin et. al.**	It utilizes CGL-GAN and ICTV for layout generation. Tagline generation utilizes multi-modal generative model. Style Attribute Predictor (SAP) has CNN based encoder and self-attention layer and cross-attention based decoder.	• Higher R1 and R1+R2 metrics than Luban and posters designed by designers.

3. REQUIREMENT ANALYSIS

3.1 Project Requirements

Operating System, GPU Server and Memory

A computer or laptop with any OS Windows, Linux, or Mac is needed to develop our system. Google Colab provides free access to GPUs, including the NVIDIA Tesla T4 with up to 16 GB of memory, which is suitable for deep learning and machine learning tasks. Kaggle also offers free access to GPUs, specifically 29 GB of 2 Tesla T4 GPUs and 1 P100 GPU, with a usage limit of 30 hours per week or 9 hours per session.

3.1.1 Software Tools and Library Requirements

VS Code

For coding tasks, Visual Studio Code (VS Code) is utilized as the primary Integrated Development Environment (IDE). The exploratory and prototyping phases of the project are enhanced by VS Code, which facilitates a thorough, team-based approach to coding. Effective prototypes are created, and iterations are carried out efficiently.

Google Colab and Kaggle

Google Colab and Kaggle are used as cloud-based platforms for executing Python code in a web-based notebook format. Free access to GPUs, including the NVIDIA Tesla T4 with up to 16 GB of memory, is provided by Google Colab, making it suitable for deep learning and machine learning tasks. However, due to insufficient GPU resources in Colab, Kaggle is also required. Free access to GPUs, specifically 29 GB of 2 Tesla T4 GPUs and 1 P100 GPU, is offered by Kaggle, with a usage limit of 30 hours per week or 9 hours per session. This ensures that the increased computational demands of the project are accommodated.

NLTK

The NLTK tokenizer is employed as a key component of Natural Language Processing to segment text into tokens, such as words, characters, or sentences. Text from English input prompts is tokenized by the system to extract key themes accurately.

HTML, CSS, and Javascript

HTML, CSS, and JavaScript are employed together to create a simple, user-friendly interface for interaction. The primary structure of the web page is created using HTML, a standard markup language. CSS is used to style HTML components, defining how web pages should be laid out. JavaScript, a scripting language, is utilized to create interactive web pages.

FastAPI

FastAPI is employed as a modern web framework for building APIs with Python 3.7+. Performance and ease of use are emphasized, with automatic interactive API documentation provided using Python type hints, ensuring user-friendliness and reducing bugs. Its combination of speed, simplicity, and powerful features makes it an excellent choice for deploying web applications and APIs.

ColorThief

ColorThief is used as a Python library to extract the single most dominant color and a palette of the most common colors in an image by processing its pixels using algorithms such as the Median Cut algorithm. Images in various formats, such as JPG and PNG, are processed by this library.

PIL/Pillow

The Python Imaging Library (PIL) is employed for opening, manipulating, and saving various image file formats. Processing capabilities like point operations, resizing, rotation, arbitrary affine transforms, drawing graphical elements, and applying filters are provided. Pillow, a fork of PIL, is utilized for backward compatibility and extended functionality.

OpenCV

The Open Source Computer Vision Library (OpenCV) is used to perform various image-related tasks. Image files are read using cv2.imread(image_path). The BGR (Blue-Green-Red) color space of an image is converted to the RGB (Red-Green-Blue) color space using cv2.cvtColor(image, cv2.COLOR_BGR2RGB). Images are displayed using the function cv2_imshow(image).

3.2 Feasibility Study

3.2.1 Technical Feasibility

Our project does not require a high degree of computer specification. Python programming will be used to create the system. The system is operational with the least number of technical requirements and hence, it is technically feasible.

3.2.2 Operational Feasibility

The Graphical User Interface (GUI) of the system will be user-friendly and interactive, making it easier to provide input prompt.

3.2.3 Economic Feasibility

Since our project is purely software-based, it is extremely feasible in terms of economy. The project will be developed on a personal computer. However, we have to buy the domain name for deploying the project. If required, we will purchase the GPU resource for training.

3.2.4 Schedule Feasibility

The project will be completed within the designated timeframe. Our project timeline is structured around key milestones, and we are prepared to address any challenges that may arise to ensure on-time completion.

3.2.5 Legal Feasibility

The dataset collection process ensures no harm is caused to any festivals or through the use of offensive language or text. Company names or logos in images will be removed during preprocessing. All datasets are sourced exclusively from public websites.

4. METHODOLOGY

This section provides the detail working of the system and the different models used.

4.1 System Architecture

This diagram shows the overall working of the system.

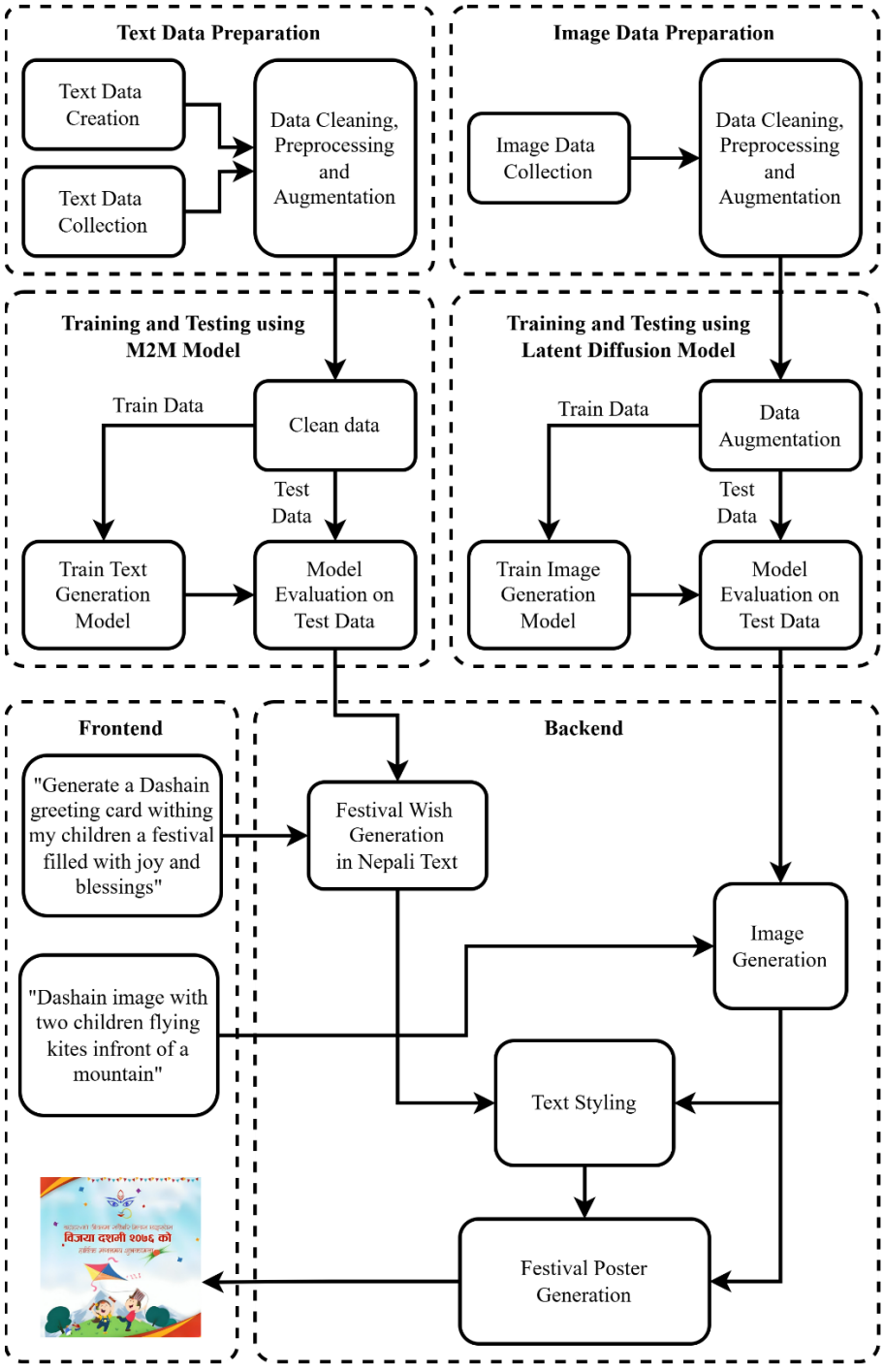


Figure 4-1: Block Diagram of System Architecture

4.1.1 Working Principle

The system asks the user to input a prompt in the English language. A fine-tuned M2M Model is used to generate festival wishes in the Nepali language. The model also extracts the festival theme from the given input prompt. Simultaneously, a fine-tuned Latent Diffusion Model is used to create a relevant image suitable for the poster based on the identified theme.

Once the image and text are generated, the ColorThief library is applied to identify dominant colors from the generated image which is then used to determine the ideal text color. At the same time, saliency mapping is employed to determine the ideal placement of styled text within the generated image. Font style is randomly generated from existing fonts available in the database. Finally, PIL, a Python Imaging Library, is utilized to seamlessly integrate the generated image with the styled text image to generate a complete festival poster which is then presented to the user.

4.2 Dataset Preparation

4.2.1 Dataset for Title Generation in Nepali Font

The dataset for title generation was collected manually from different sources, such as social media posters, greeting cards, festival messages, and other relevant Nepali literature. The dataset contains text prompt in English and its corresponding wishes or greetings text in Nepali font.

For example:

Prompt: "I want a poster to wish happy Dashain to Nepali people for the year 2079."

Text: "२०७९ सालको दशैंको हार्दिक मंगलमय शुभकामना!"

This pair was stored in a structured format such as a CSV file for easy access under two labels ‘Prompt’ and ‘Text’. The text prompt as input was cleaned by removing stop words and tokenized to create a cleaned dataset. Finally, the data was ready to be fine-tuned on a pre-trained M2M Model.

4.2.2 Dataset for Image Generation in Nepali Font

The project aims to create a dataset for training a title generation model focused on five Nepali festivals: Dashain, Tihar, Chhath, New Year, and Holi. The dataset was manually collected from various websites like Pinterest, Facebook, and Dribble, featuring festival-related images with styled Nepali text. Before inclusion, all non-essential elements like text, logos, and other information were removed using inpainting tools. The image resolution was then enhanced to ensure high quality.

To augment the dataset and improve model training, various techniques were applied. For each image in the dataset, a corresponding prompt was written that highlighted the elements present in the image.

For example,

Prompt: “A Tihar image of woman carrying diyos with mandalas, flowers, lights and fireworks in the background”

Image:

Figure 4-2: Example of Image of Image Dataset

CSV

The prompt was stored in a csv file along with the image name, under “Prompt” and “Images” respectively, which was essential for training generative models effectively, enabling them to generate accurate and contextually appropriate titles for festival-related images. Figure 4-3 shows the .....

(a) Raw Image (b) Text Removed Image (c) Enhanced Image

Figure 4-3: Sample Images of Dataset Preparation Steps for Tihar Festival

4.3 Title Generation

Title generation is the first and crucial phase of our system. This stage ensures the final poster conveys the intended message and sentiment appropriately. Firstly, the user enters the prompt in English for generating festival posters. The input prompt describes the desired message or wish for a specific festival. Once the prompt is entered, our system processes this input to generate a coherent, logical, and well-contextualized Nepali title or wish that will be kept in the poster. This involves understanding the context of the input prompt by identifying the festival and extracting the key message or wish.

The system uses the M2M-100 model to generate wishes in Nepali language based on the input prompt.

4.3.1 M2M-100 Architecture

M2M-100 is a multilingual sequence-to-sequence model trained for Many-to-Many multilingual translation. It is composed of two modules: 12-layer encoder and 12-layer decoder, both utilizing Transformer layers that include self-attention and feed-forward networks. The encoder processes the source language input, converting it into high-dimensional embeddings that capture the sentence's semantic meaning. The decoder then generates the target language output, leveraging cross-attention to focus on

Use Figure 4-4 info to explain this arch. more clearly

relevant parts of the encoded input while maintaining coherence through self-attention. It contains 8192 Feed-Forward-Network size and 1024 embedding dimensions. The total parameter count is 1.2B.

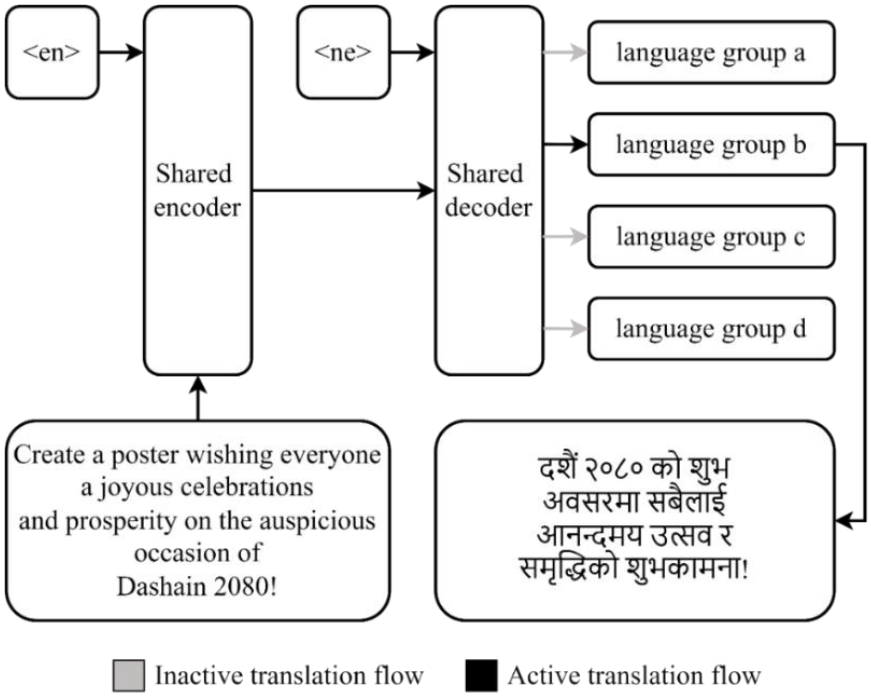


Figure 4-4: M2M Model Architecture (English to Nepali)

Encoder

The encoder in the M2M-100 model is responsible for converting the input sequence in the source language into a high-dimensional representation that captures the semantic meaning of the sentence. The encoder consists of 12 layers of Transformer blocks, each containing several key components.

Make Bold

Scaled Dot-Product Attention: This mechanism computes attention scores for each word in the input sequence with respect to all other words. The attention scores are then used to create a weighted sum of the input embeddings, highlighting the most relevant words for each position in the sequence.

$$\text{Attention}(\text{Q}, \text{K}, \text{V}) = \text{softmax}\left(\frac{\text{QK}^{\text{T}}}{\sqrt{\text{d}_\text{k}}}\right)\text{V} \qquad 4.1$$

where Q (Queries), K (Keys), and V (Values) are derived from the input embeddings, and d_k is the dimensionality of the keys.

bold

Multi-Head Attention: Instead of a single attention function, multiple attention heads are used to allow the model to focus on different parts of the input sequence simultaneously. Each head performs the attention operation independently, and their outputs are concatenated and linearly transformed.

$$\text{Concat}(\text{Q}, \text{K}, \text{V}) = \text{Concat}\,(\text{head}_1, \text{head}_2, \ldots, \text{head}_{\text{h}})\text{W}^{\text{o}} \quad 4.2$$

where each head is an independent attention mechanism and W^o is a weight matrix.

Feed-Forward Network: Each position in the input sequence is processed independently through a two-layer MLP with a ReLU activation function between the layers.

$$\text{FFN}(\text{x}) = \max(0, \text{x}\text{W}_1 + \text{b}_1)\,\text{W}_2 + \text{b}_2 \quad 4.3$$

where, x is Input to the feed-forward network, W_1 is Weight matrix of the first layer, b_1 is Bias of the first layer, W_2 is Weight matrix of the second layer, and b_2 is Bias of the second layer

Residual Connections and Layer Normalization: Skip connections add the input of each sub-layer to its output, facilitating gradient flow and enabling the training of deeper networks.

For self-attention:

$$\text{Z} = \text{norm}\big(\text{X} + \text{self} - \text{attention}(\text{X})\big) \quad 4.4$$

where, X is input to the first self-attention layer and $self - attention(X)$ is self-attention applied to X.

For Feed-Forward Network:

$$\text{Y} = \text{norm}\big(\text{Z} + \text{FFN}(\text{Z})\big) \quad 4.5$$

where, Z is output after the first self-attention and $FFN(Z)$ is feed-forward network applied to Z.

Encoding Process: The encoder takes the sequence of tokens $W = (w_1, \ldots w_s)$ as input along with the source language l_s . The input sequence and source language is tokenized into high-dimensional vectors. These embeddings are then passed through the 12 Transformer layers, resulting in a sequence of embeddings $H = (h_1 \ldots h_s)$ that encapsulates the semantic content of this input sequence.

$$\mathrm{H} = \mathrm{encoder}(\mathrm{W}, \mathrm{l_s}) \qquad 4.6$$

Decoder

The decoder generates the target language output sequence from the high-dimensional representations produced by the encoder. The previously created tokens are first turned into embeddings and sent through the decoder. The decoder looks at earlier tokens to understand the context and then uses the encoder's output to incorporate information from the original sentence. This combined information is processed to refine the hidden states. Finally, these hidden states are used to predict the next token by passing them through a layer that applies softmax activation, giving a probability distribution over possible next words. Like the encoder, the decoder also consists of 12 layers of Transformer blocks, with additional mechanisms to handle the sequential nature of output generation.

bold

Self-Attention Mechanism: Similar to the encoder's self-attention, but with a causal mask to ensure that each position in the sequence can only attend to previous positions. This prevents the model from "seeing" future tokens during training and inference.

$$\mathrm{MaskedAttention}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \mathrm{softmax}\left(\frac{\mathrm{QK^T}}{\sqrt{\mathrm{d_k}}} + \mathrm{mask}\right)\mathrm{V} \qquad 4.7$$

where Q (Queries), K (Keys), and V (Values) are derived from the input embeddings,

d_k is the dimensionality of the keys, and $mask$ is a Causal mask.

Cross-Attention Mechanism: This mechanism allows the decoder to attend to the encoder's output embeddings H, effectively linking the input and output sequences. The attention scores are computed between the decoder's current hidden states and the encoder's output.

$$\text{MaskedAttention}(\text{Q}, \text{K}, \text{V}) = \text{softmax}\left(\frac{\text{QH}^{\text{T}}}{\sqrt{\text{d}_\text{k}}}\right)\text{H} \quad 4.8$$

where Q (Queries), H is encoder's output and d_k is the dimensionality of the keys.

Multi-head attention is used here as well, enabling the decoder to focus on different aspects of the encoder's output.

Feed-Forward Network (FFN): Similar to the encoder, the decoder also uses a two-layer MLP with ReLU activation to process the hidden states from the attention layers.

Residual Connections and Layer Normalization: Applied in the same way as in the encoder, facilitating gradient flow and model training.

$$\text{Z} = \text{Norm}\big(\text{Y} + \text{CrossAttention}(\text{Y}, \text{H})\big) \quad 4.9$$

$$\text{Y} = \text{Norm}(\text{Z} + \text{FFN}(\text{Z})) \quad 4.10$$

where, Y is an input to the Cross-Attention mechanism, Z is an output of the Cross-Attention mechanism, and $Norm$ is Layer normalization function.

Decoding Process: The decoder takes the sequence of embeddings of H with the target language l_t and sequentially produces the target sentence token by token $V = (v_1, \dots v_s)$. The decoding process is repeated until the entire target sentence is generated.

$$\text{v}_{\text{i}+1} = \text{decoder}(\text{H}, \text{l}_\text{t}, \text{v}_1 \dots \text{v}_\text{i}) \quad 4.11$$

Vocabulary Construction

M2M-100 model has been designed for the bilingual case, where the target language is fixed. In the case of multilingual machine translation, the target language is not fixed, so they added a special token in the encoder indicating the source language l_s and a special token in the decoder indicating the target language l_t.

Additionally, it uses shared vocabulary that allows it to learn cross-lingual representations. This shared vocabulary is typically constructed using techniques like

Byte Pair Encoding (BPE) or SentencePiece to handle the diverse character sets and token frequencies of different languages.

Use Figure 4-5 to describe relevant topic

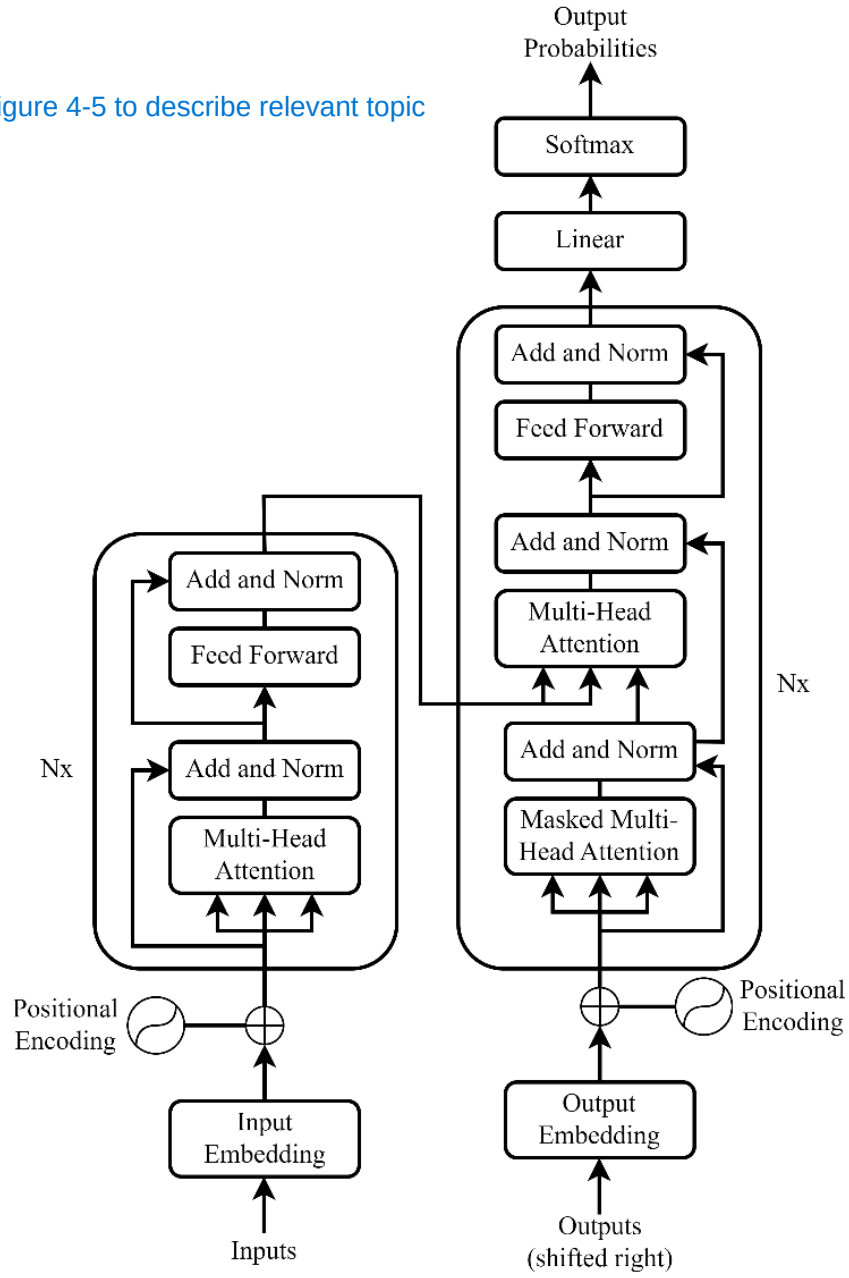


Figure 4-5: Transformer Model Architecture

Byte Pair Encoding (BPE): BPE creates a shared subword vocabulary across languages by breaking down rare words into common subword units, enabling the model to learn cross-lingual representations more effectively. By employing BPE, the M2M-100 model manage vocabulary size by breaking down rare words into common subword units. This allows the model to handle diverse languages and morphologically rich

languages by reusing subword units across different words, thus improving generalization and reducing the number of out-of-vocabulary words.

SentencePiece: SentencePiece operates directly on raw text and generates subword units, which can handle rare and unknown words. It supports unsupervised training on raw text, making it adaptable to various languages and scripts. It includes text normalization and customizable vocabulary size, enhancing the model's ability to process multiple languages efficiently. This flexibility helps manage the diverse character sets and word frequencies of different languages. The customizable vocabulary size allows the M2M-100 model to balance vocabulary coverage and model complexity, optimizing performance for different language pairs.

4.3.2 Training Details

The model is trained on a vast multilingual dataset that includes parallel sentences from many language pairs with the Adam optimizer and warmed-up first for 4000 updates, with label smoothing 0.1. For regularization, the dropout parameter is tuned between {0.1, 0.2, 0.3}. To stabilize the training of deeper Transformers, it is trained with LayerDrop 0.05 and pre-normalization.

4.3.3 Evaluation Metrics

BLEU Score

BLEU, or the Bilingual Evaluation Understudy, is a score for comparing a candidate translation of text to one or more reference translations. It measures the overlap between n-grams in the generated text and reference text, indicating how similar the generated text is to the reference text.

The formula for BLEU score is as follows:

$$\text{BLEU} = \text{BP} \times \exp\left(\sum \text{p}_n\right) \tag{4.12}$$

where, BP(Brevity Penalty) is a penalty term that adjusts the score for translations that are shorter than the reference translations and pn is the precision of n-grams.

BLEU score ranges from 0 to 1, with higher values indicating better translation quality.

METEOR score

The METEOR score is a metric for evaluating machine translation quality that emphasizes alignment between the translation and the reference. It incorporates exact matches, stem matches, synonym matches, and paraphrase matches, offering a comprehensive evaluation. The score is derived from a harmonic mean of precision and recall, further adjusted by a fragmentation penalty to reflect alignment coherence.

The formula for Meteor score is as follows:

$$\mathrm{M} = \mathrm{F}_{\mathrm{mean}}(1 - \mathrm{Penalty}) \tag{4.13}$$

where,

F_{mean} combines precision (P) and recall (R) with a weight parameter α (usually set to 0.9) and is given by:

$$\mathrm{F}_{\mathrm{mean}} = \frac{\mathrm{P.R}}{\propto.\mathrm{p}+(1-\propto).\mathrm{R}} \tag{4.14}$$

Similarly, Fragmentation Penalty ($Penalty$) lowers the score for translations with many separate matched segments, encouraging more continuous and coherent translations and is calculated as:

$$\mathrm{Penalty} = \gamma\left(\frac{\mathrm{C}}{\mathrm{m}}\right)^{\beta} \tag{4.15}$$

Here, C is the number of chunks (continuous matched segments), m is the number of matched words, γ is a scaling factor (typically set to 0.5), β and is an exponent (typically set to 3).

ROUGE

ROUGE score measures the similarity between the machine-generated summary and the reference summaries using overlapping n-grams, word sequences that appear in both the machine-generated summary and the reference summaries, focusing on recall and precision of n-grams. The range for ROUGE metrics is from 0 to 1, where 0 represents no overlap or similarity, and 1 represents perfect overlap or similarity.

The formula for ROUGE score is as follows:

$$\text{ROUGE} - \text{N} = \frac{\sum_{S \in \{\text{ReferenceSummaries}\}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S \in \{\text{ReferenceSummaries}\}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)} \quad 4.16$$

ROUGE-1 measures the overlap of unigrams (single words) between the generated summary and the reference summaries. It provides a basic measure of how many words in the generated summary are present in the reference summaries

ROUGE-2 measures the overlap of bigrams (pairs of consecutive words) between the generated summary and the reference summaries.

ROUGE-L measures the longest common subsequence (LCS) between the generated summary and the reference summaries. It accounts for the longest sequence of words that appear in both the generated and reference summaries in the same order.

$$\text{ROUGE} - \text{L} = \frac{\text{LCS(generated,reference)}}{\text{Length(reference)}} \quad 4.17$$

Self-BLEU

Self-BLEU is a metric to measure the diversity of generated texts. The formula involves treating each generated text in a set as a reference and calculating BLEU scores against the rest of the generated texts:

It is calculated as:

$$\text{self} - \text{BLEU} = \frac{1}{\text{N}} \sum_{\text{i}=1}^{\text{N}} \text{BLEU}(\text{g}_\text{i}, \text{G} \backslash \{\text{g}_\text{i}\}) \quad 4.18$$

where, $G = \{g_1, g_2, \ldots., g_N\}$ is the set of generated texts, and BLEU $(g_i, G\backslash\{g_i\})$ is the BLEU score of g_i against the rest. ???? self-BLEU scores will range between 0 and 1, where a lower score indicates higher diversity among the generated texts.

4.4 Image Generation

Once the theme is identified from the input prompt, a new image specific to that theme is generated using a pre-trained Latent Diffusion Model. This model, which has set a new state-of-the-art score for class-conditioned image generation and image inpainting,

produces high-quality samples with increased diversity. The traditional diffusion models generate high-quality samples with more diversity but they are slow in sampling and require large computational resources. Unlike traditional diffusion models that work directly in pixel space, the Latent Diffusion Model first encodes high-resolution image data into a fixed latent representation, thereby reducing computational complexity.

4.4.1 Latent Diffusion Model

Generating images with the Latent Diffusion Model involves iterative denoising noise samples to produce high-resolution images that closely resemble the training data. The model conditions this image generation process using input prompts. Training the model involves two main steps: first, training an autoencoder comprising an encoder and a decoder. Once trained, the autoencoder does not need further training in the subsequent diffusion model phase. The autoencoder learns to encode any input data into a consistent latent representation of 256x256 dimensions while preserving the original data's features. Following this, the diffusion model is trained to denoise the noise samples, a process akin to other diffusion models.

The Latent Diffusion Model benefits from cross-attention mechanisms, which aid in generating images based on input prompts. Various methods can be employed to extract information from inputs, whether they are text or image-based. For textual inputs, the BERT tokenizer is used to tokenize the input data, integrating it into a UNet layer using a cross-attention mechanism.

Mention Figure4-6 to explain some part of this section

Previously, diffusion models operated directly in pixel space, requiring significant computational resources and repeated denoising of noise samples. In contrast, Latent Diffusion Models incorporate a compressive phase. This phase includes an autoencoding model that learns a lower-dimensional representation of input data, maintaining perceptual equivalence with the original image space. The generative phase resembles traditional diffusion models, generating outputs in latent space, which are then converted into high-dimensional space in a single step.

The proposed approach offered several advantages:

- The computational complexity is significantly reduced by operating in a low-dimensional space learned by an autoencoder. The autoencoder is trained once then it can be used to encode the data into latent space.
- Using the built-in assumptions (Inductive bias) of UNet architecture in diffusion models improves their performance, especially for data with spatial patterns.
- The proposed method yields general-purpose compression models with a latent space for multiple purposes. This latent space can be used to train diverse generative models and for various downstream applications, including tasks like single-image CLIP-guided synthesis.

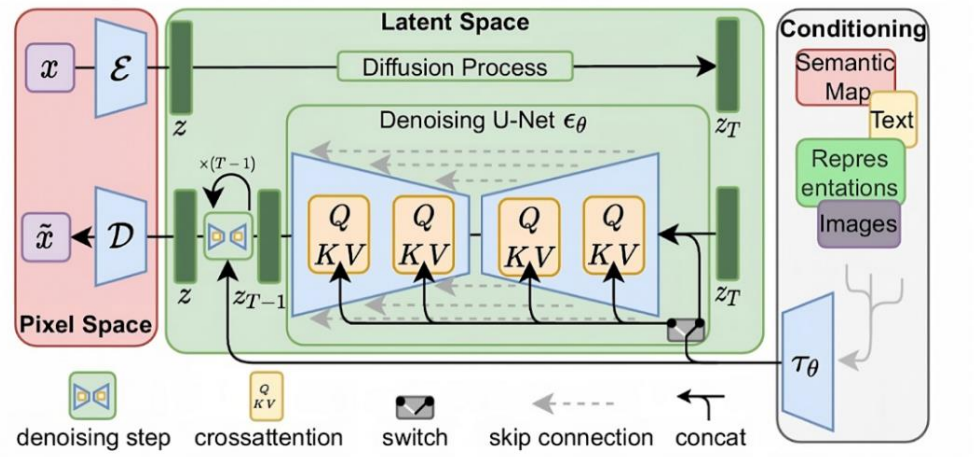


Figure 4-6: Architecture of Latent Diffusion Model

Perceptual Image Compression

The perceptual image compression has an autoencoder. Autoencoder consists of encoder and decoder architectures. The Latent Diffusion model uses an autoencoder to represent the image in a latent space for easy computation.

Given an input image x ∈ $\mathbb{R}^{H\times W\times 3}$ in RGB space:

Here, H represents height, W represents weight and 3 is the RGB value for each pixel.

Encoder (E) encodes x into a latent representation z.

$$z = E(x) \tag{4.20}$$

Decoder (D) reconstructs the image from the latent representation.

$$\bar{x} = D(z) = D\big(E(x)\big) \tag{4.21}$$

where $z \in \mathbb{R}^{H \times W \times 3}$

The encoder downsamples the image by dividing it by $f = \frac{H}{h} = \frac{W}{w}$. Different downsampling factors $f=2^m$ with $m \in \mathbb{N}$ are explored.

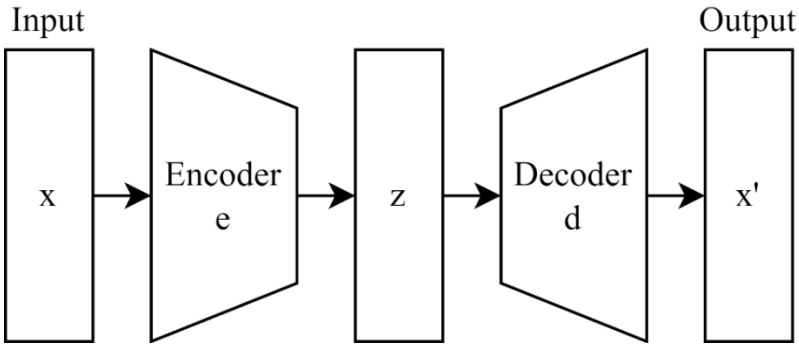


Figure 4-7: Encoder Decoder Architecture for Latent Representation

First, the autoencoder is trained to generate a latent representation that preserves the image features even in latent space. To avoid high-variance latent spaces, two types of regularizations are experimented with KL Regularization (KL-reg.) and Vector Quantization Regularization (VQ-reg.). They experimented with the generated quality using different downsampling factors and concluded that downsampling factors 4 and 8 outperform models.

KL-Regularized Latent Space

KL Regularization (KL-reg.) applies a slight KL-divergence penalty towards a standard normal distribution on the learned latent space, similar to a Variational Autoencoder (VAE). If the output of the encoder for input x is E(x), then the sampled latent space is defined as $z = E(x) + \epsilon$, where $\epsilon \sim N(0,1)$ represents noise sampled. After sampling z, it undergoes resampling to a unit standard deviation as $\bar{z} = \frac{z}{\sigma}$ where, σ is standard deviation.

VQ-Regularized Latent Space: Vector Quantization Regularization (VQ-reg.) incorporates a vector quantization layer within the decoder, analogous to VQGAN but with the quantization layer integrated into the decoder. But, VQ-regularized latent spaces give a better quality of the generated image. Before feeding z into the decoder D, it undergoes quantization. Quantization is the process of mapping each element of z to the nearest codebook vector from a predefined set of vectors. Then, this quantized z is feed into decoder D.

Losses in Autoencoder

Extra Space?

Autoencoder loss is a combination of 3 losses: Reconstruction, Adversarial, and Regularization loss.

$$L_{Autoencoder} = \min_{\varepsilon,D} \max_{\varphi} \left(L_{rec}(x, D(\varepsilon(x)) - L_{adv}\left(D\left(\varepsilon(x)\right)\right) + \log D_{\varphi}(x) + L_{reg}(x; \varepsilon, D) \right) \quad 4.22$$

Perceptual Loss or Reconstruction loss

Extra space?

Perceptual loss is a loss function that measures the difference in the high-level features. These features are extracted from a pre-trained neural network (usually a convolutional neural network like VGG).

$$\text{Pertpetual Loss} = (\text{MSE}(x, D\left(E(x)\right)$$

$$= 1\frac{1}{N}\sum_{i=1}^{n} \|\emptyset_i(x) - \emptyset_i D(E(x))\|_2^2 \quad 4.23$$

where, x is the input image, D(E(x)) is the output image from the decoder and i are the feature maps from layer i of a pre-trained network (like VGG), and N is the number of layers. This evaluates whether the input image is the same as the output image from the decoder.

Regularization Loss

$$\text{Regularization loss (Lreg)} = \text{KL}(E(x) = L || N(0,1)) \quad 4.24$$

It is just a form of normalization in latent space, where $E(x) = L$ represents latent space which is the output of the encoder. This evaluates whether the latent space is normalized

or not. || is the symbol for Kullback-Leibler (KL) divergence between two probability distributions i.e. $E(x)$ and $\mathrm{N}(0,1)$.

$$\text{Regularization loss (Lreg)} = \sum \mathrm{E(x)} \log\left(\frac{\mathrm{E(x)}}{\mathrm{N(0,1)}}\right)$$

$$= -\sum \mathrm{E(x)} \log\left(\frac{\mathrm{E(x)}}{\mathrm{N(0,1)}}\right) \qquad 4.25$$

Patch-based Adversarial Objective

The patch-based adversarial objective is used in GANs to ensure local realism in generated images. It divides the image into smaller patches, and a discriminator evaluates each patch independently.

$$\text{Adversarial Loss}(\mathrm{L_{adv}}) = \log(1 - \mathrm{D_{\varphi}}\left(\mathrm{D}\left(\mathrm{E(x)}\right)\right) + \log(\mathrm{D_{\varphi}}(\mathrm{D(x)}) \qquad 4.26$$

Or

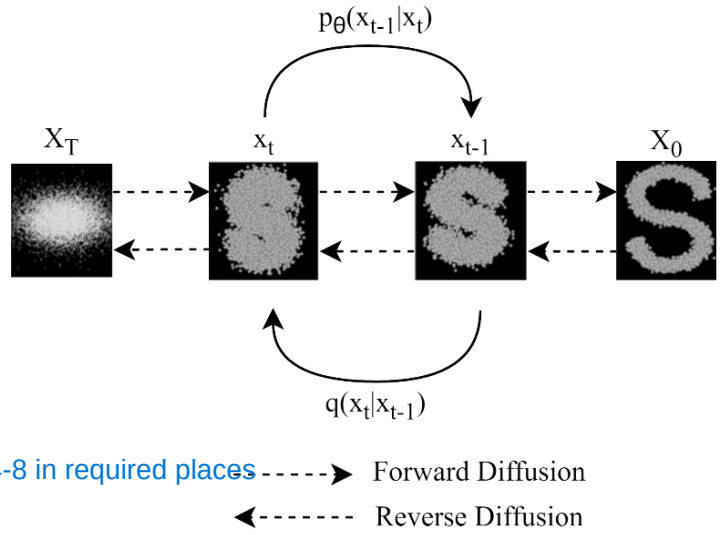
$$\text{Adversarial Loss}(\mathrm{L_{adv}}) = -\log(1 - \mathrm{D_{\varphi}}\left(\mathrm{D}\left(\mathrm{E(x)}\right)\right) + \log(\mathrm{D_{\varphi}}(\mathrm{D(x)}) \qquad 4.27$$

This loss goal is to maximize so that the generated image is as real as the input image. This is evaluated by discriminatorD_{φ}. The $D_{\varphi}(x)$ evaluates if the input image is real or not so it wants to maximize it. The goal of this loss is to train the discriminator D to maximize log (D_{φ} (x)) i.e. to classify the input image as real correctly and to maximize $\log(1 - \mathrm{D_{\varphi}}\left(\mathrm{D}\left(\mathrm{E(x)}\right)\right)$ (or minimize $-\log(\mathrm{D_{\varphi}}(\mathrm{D(x)})$ i.e. to classify the denoted image as fake correctly.

Using all these loss functions to train the autoencoder ensures that the latent representations preserve important features both when encoding images into latent space and when decoding latent representations back into images.

Latent Diffusion Model

Diffusion models are probabilistic models that learn how to generate data by gradually improving noisy data. In the Latent Diffusion Model, the diffusion process is the same but this process is performed in latent space which makes the model suitable for high-resolution images.



Mention Figure 4-8 in required places

Figure 4-8: Working of Diffusion Model

There are two steps in training a diffusion model: forward and backward diffusion.

Forward Diffusion

First, these models generate noisy data from the original data gradually step by step. This process is usually predefined and does not involve training. The noise is added in a specific pattern or rule (Markov Chain), where each step depends on the previous one. The noise added denoted as z is in normal distribution i.e. $z \sim N(0,1)$. After many steps (usually 1000-5000 steps) the data is purely noisy.

After obtaining noisy data, the model is then trained to denoise the noisy data into the original image. This way the model learns to generate images from noisy data. This process also takes several steps just like the forward diffusion process and follows the Markov Chain rule. Since this takes time and computational resources, latent space is used instead of high-dimensional data. The main task of the diffusion model is to denoise the noisy data which requires training. For this, Time-Conditioned UNet architecture is used. Time-conditioned UNet is a convolutional neural network architecture that was developed for image segmentation tasks with additional information (conditions). This convolutional neural network is designed to learn from fewer training samples. This additional information guides the segmentation task. UNet consists of a sequence of denoising autoencoders $\epsilon\theta(xt, t)$ for generating a cleaner image.

Diffusion models in terms of Signal-to-noise-ratio

Diffusion models can be specified in terms of a signal-to-noise ratio $\mathrm{SNR(t)} = \frac{\alpha_t^2}{\sigma_t^2}$ consisting of signal α_t that we want to extract during the diffusion process and the noise σ_t present in the data where t ranges from 1 …. T. Here,

For a data sample x0, the q in forward diffusion process is defined as

$$\mathrm{q(x_t|x_o)} = \mathrm{N(x_t|\alpha x_o, \sigma_t^2 I)} \quad 4.28$$

where I is the identity matrix.

With the Markov structure for s<t:

$$\mathrm{q(x_t|x_o)} = \mathrm{N(x_t|\alpha_{t|s} x_{t|s}, \sigma_{t|s}^2 I)} \quad 4.29$$

$$\alpha_{t|s} = \frac{\alpha_t}{\alpha_s} \quad 4.30$$

$$\alpha_{t|s} = \alpha_{t|s}^2 - \alpha_{t|s}^2 \sigma_s^2 \quad 4.31$$

The goal of denoising diffusion models $p(x_0)$ is to reverse the diffusion process. $\mathrm{p(x_0)}$ is the probability distribution of the initial original data $\mathrm{x_0}$.

$$\mathrm{p(x_0)} = \int \mathrm{p(x_T)} \quad 4.32$$

So, denoising the data sample to original data is done using

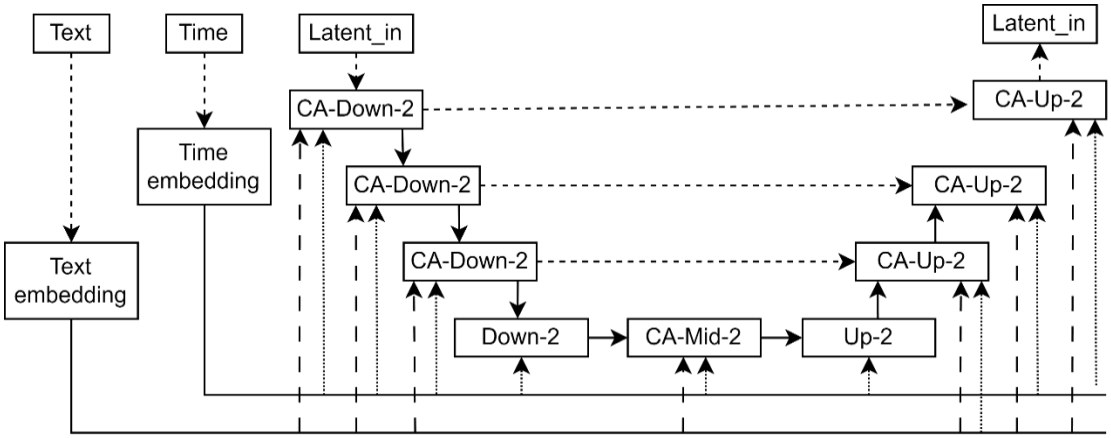
$$\mathrm{p(x_{t-1}|x_t)} = \mathrm{q\left(x_{t-1}\middle|x_t, x_\theta(x_t, t)\right)} = \mathrm{N\left(x_{t-1}|\mu_\theta(x_t, t), \sigma_{t|t-1}^2 \frac{\sigma_{t-1}^2}{\sigma_t^2} I\right)} \quad 4.33$$

where the mean can be expressed as

$$\mu_{\theta(x_t, t)} = \frac{\alpha_{t|t-1}\sigma_{t-1}^2}{\sigma_t^2} x_t + \frac{\alpha_{t|t-1}\sigma_{t-1}^2}{\sigma_t^2} x_\theta(x_t, t) \quad 4.34$$

Also, is the output of the conditional denoising autoencoder dependent on x_t , diffusion step t.

$$\epsilon_\theta(x_t, t) = (x_t - \alpha_t x_\theta(x_t, t))/\sigma_t \quad 4.35$$



Citation needed?

Figure 4-9: Time Conditioned UNet Architecture

Mention Figure 4-9 in required places

Conditioned-UNet has following parts:

Encoder-decoder structure

The figure shown is a U-shaped encoder-decoder architecture. There are four encoder blocks on the left side and four decoder blocks on the right. Each encoder and decoder block contains convolution using a 3x3 matrix and then uses the ReLU activation function for non-linear data.

Encoder block: Encoder downsampled the data to capture high-level features i.e. reduces height and weight. This is shown using a Down-2 block. The CA-Down-2 block represents downsampling with a cross-attention block.

Middle block: The CA-Mid-2 is the intermediate output of UNet which contains data with the smallest height and weight and most channels.

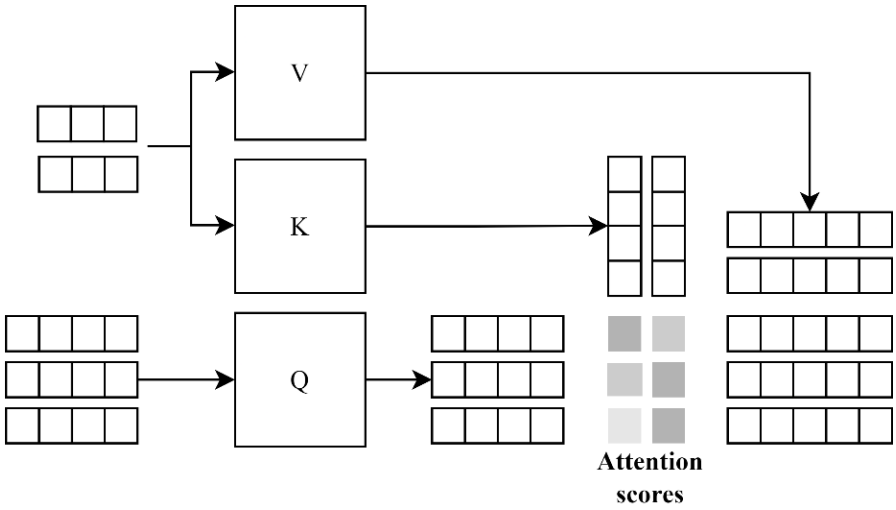
Decoder block: Then, the decoder upsamples into the original data. This is shown using an Up-2 block. The connection between CA-Down-2 and CA-Up-2 is a skip connection. Skip connections take the output of CA-Down-2 directly into CA-Up-2 which performs upsampling that helps to retain high-resolution data. The CA-Up-2 block represents upsampling with a cross-attention block. At last, a convolution using a 1x1 matrix is performed which gives an output image having dimension hxwxd; d is the number of defined segment classes.

Conditioning Mechanisms

To make the diffusion models more flexible in generating conditional images, a cross-attention mechanism is integrated into the UNet backbone. This mechanism enhances the model's ability to handle text prompts. Text and time embedding is an important feature for conditioned image generation.

Text Embedding: For text-conditioned image generation, the embedding of input text is passed into each encoder and decoder block. A pre-trained language model i.e. BERT is used to convert the input text into a dense vector representation. This text embedding captures the semantic meaning of the text. These text embeddings are concatenated with the feature maps at different encoder and decoder blocks.

Time Embedding: Time is a crucial part of the Latent Diffusion model which takes into account sequential data during the denoising process. Time embedding is used at every part of the diffusion process and passed to the encoder-decoder layer. Time embedding provides the model with information about the specific time step in the diffusion process. Also, the model can appropriately adjust its parameter by knowing the current timestep t to effectively denoise the latent representation at each stage.



Same for this figure

Figure 4-10: Cross Attention Mechanism

Cross Attention Mechanism

For cross attention mechanism, queries, keys, and values are used which are extracted from the text and image. Queries represent what the model focuses on. Keys are

representations of the available context (e.g. conditioning embeddings). Values are the actual information or content associated with each key.

Given z as the image input, and y be the conditioning input (e.g., text embeddings, time embeddings).

$$\text{Queries (Q)} = z_t \cdot W_Q \tag{4.36}$$

$$\text{Keys (K)} = y_t \cdot W_K \tag{4.37}$$

$$\text{Values (V)} = y_t \cdot W_V \tag{4.38}$$

where, W_Q, W_K, and W_V are the learnable weights for queries, keys, and values.

Once Q, K, and V are computed, the cross-attention mechanism calculates the attention weight as

$$\text{Attention weights } (\alpha) = \text{softmax}(QK^T/\sqrt{d}) \tag{4.39}$$

Then a score is calculated as

$$\text{Score} = \alpha V = \text{softmax}(QK^T/\sqrt{d}).V \tag{4.40}$$

This score provides a weighted sum that emphasizes relevant information based on the queries and gives the probability of embedding for the query.

4.4.2 Loss in the Latent Diffusion Model

Based on pairs of images and conditioning inputs, we train the conditional Latent Diffusion Model (LDM) using the following objective:

$$L_{LDM} := E_{E(x,y),\epsilon\sim N(0,1),t} - \left[\left\|\epsilon - \varepsilon_\theta(z_t, t, \mathcal{T}_\theta(y))\right\|_2^2\right] \tag{4.41}$$

where is the noise sampled from a standard normal distribution i.e. $\sim N(0,1)$, is the output of the conditional denoising autoencoder dependent on zt, diffusion step t, and the intermediate representation $\mathcal{T}_\theta(y)$ of the conditioning input y.

Both $\mathcal{T}_\theta$ and ε_θ are optimized together according to this objective. The conditioning mechanism is adaptable, allowing $\mathcal{T}_\theta$ to be parameterized with domain-specific models like transformers, particularly useful when conditioning inputs y are text prompts.

4.4.3 Training parameters

LDM-4 is considered more optimized and better suited compared to earlier versions of LDMs. High-quality images are generated that align perfectly with the objectives. Therefore, LDM-4 is required.

Table 4-1: Hyperparameters for LDMs Trained on ImageNet Dataset

Parameter	**Values**
z-shape	64×64×64
|Z|	8192
Diffusion steps	1000
Noise Schedule	Linear
Model Size	319M
Channels	192
Depth	2
Channel Multiplier	1,2,3,5
Number of Heads	1
Batch Size	40
Iterations	2M
Learning Rate	82-5
Conditioning	CA
CA-resolutions	32,16,8
Embedding Dimension	512
Transformer Depth	1

4.4.4 can be started from here.

4.4.4 Evaluation Metrics

Frechet Inception Distance (FID)

The Frechet Inception Distance score is a metric that calculates the distance between feature vectors calculated for real and generated images. Lower FID scores indicate that the generated images are closer to the real images in terms of distribution, which typically corresponds to higher quality and more realistic generated images.

Let (M_t, C_t) and (M_g, C_g) represent the mean and covariance of the true and generated features respectively, then we compute

$$\mathrm{FID} = \left\|M_t - M_g\right\|_2^2 + \mathrm{Tr}\left(C_t + C_g - 2\left(C_t C_g\right)^{\frac{1}{2}}\right) \qquad 4.42$$

Qualitative Metrics

Qualitative metrics are descriptive indicators based on subjective assessment.

Human Evaluation: Subjective assessment by human evaluators on criteria such as fluency, coherence, and relevance of the generated text.

Aesthetics Score: Volunteers or experts rate the quality, relevance, and aesthetics of the generated posters and how well the generated layouts match the provided textual descriptions. Volunteers score the generated posters on a certain scale on the aesthetics of the layouts then the mean and the standard deviation of the scores are taken.

4.5 Text Styling

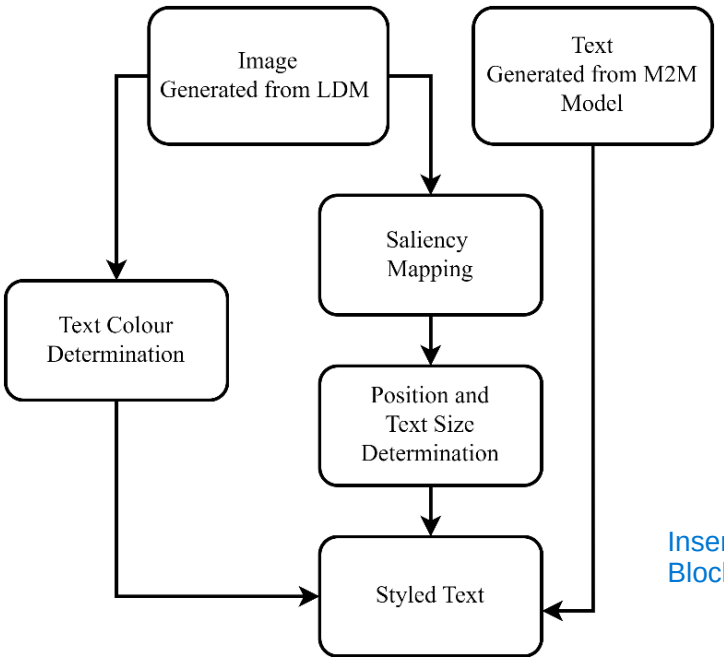
The festival wish in the Nepali language generated by the M2M model is transformed into an aesthetically pleasing styled text using a text styling algorithm. The algorithm determines the font type, font colour, font size, and position of text. ColorThief, a Python package is used to extract the dominant color i.e. background color and six best palettes from the image generated by the LDM. The colours are then used to determine the text colour by ensuring there is enough contrast between the text colour, which is randomly extracted within the image colour palette, and the background colour. The text font type is selected randomly from a collection of fonts stored in the database. Saliency mapping technique is used to determine font size and text position on the

refer explained section

Explain dominant color extraction principle using C.T. library

Mention Figure 4-11

festival post. Once all the features are determined, a styled image picture is generated which is to be placed on the background image.



Insertion of Text in Image Block??

Figure 4-11: Text Styling Procedure for Generated Text

4.5.1 Modified Median Cut Quantization

The modified median cut algorithm for color quantization involves partitioning the color space into a small number of regions, each represented by a single color. This process reduces the number of colors in an image while maintaining visual fidelity. The algorithm iteratively divides the color space into rectangular regions (vboxes) with roughly equal numbers of pixels. The key steps include choosing vboxes for subdivision, selecting the axis along which to divide, and determining the division point.

Algorithm for Modified Median Cut Quantization (MMCQ) Use standard algo format

STEP-1: Initialize:

1.1. Set the initial vbox to cover the entire RGB color space.

1.2. Build the histogram of colors from the image.

STEP-2: First Stage of Subdivision:

2.1. Select a fraction f of the vboxes based on pixel population.

2.1.1. Sort vboxes by their population.

2.1.2. Select the top f fraction of vboxes for subdivision based on population.

2.2. For the remaining fraction $1 - f$,

2.2.1. Select vboxes based on the product of vbox volume and pixel population.

2.2.2. Sort vboxes by the product of vbox volume and pixel population.

2.2.3. Select the top $1 - f$ fraction of vboxes for subdivision based on the product.

STEP-3: Subdivide Each Vbox:

3.1. Determine the axis for subdivision:

3.1.1. Choose the largest axis of the vbox.

3.2. Locate the split point:

3.2.1. Find the median pixel by population along the chosen axis.

3.2.2. Divide the vbox at the midpoint of the side containing the median pixel.

3.3. Ensure effective subdivision:

3.3.1. Avoid placing low-density clusters in the same vbox as high-density clusters.

STEP-4: Generate Colormap:

4.1. For each vbox, compute the average color within the vbox.

4.2. Create a colormap where each entry represents the average color of a vbox.

STEP-5: Convert Histogram to Inverse Colormap:

5.1. For each quantum volume in the histogram, map it to the index of the corresponding vbox in the colormap.

STEP-6: Quantize the Image:

6.1. For each pixel in the image:

6.1.1. Look up the nearest color in the colormap using the inverse colormap.

6.1.2. Replace the pixel's RGB value with the nearest colormap index.

STEP-7: Handle Dithering:

7.1. If dithering is used, ensure the initial vbox includes the entire RGB space.

7.2. Cap dithering oscillations by limiting color transfer for each component.

STEP-8: Finalize to check for fidelity and minimize visible artifacts, especially in nearly constant color regions.

4.6 Text-Image Integration

The background image, generated by the LDM, and the styled text image, generated using a text styling algorithm, are integrated into a single festival post using the PIL library. The styled text is placed at the position determined by saliency mapping technique.

Explain with the help of Figure 4-12

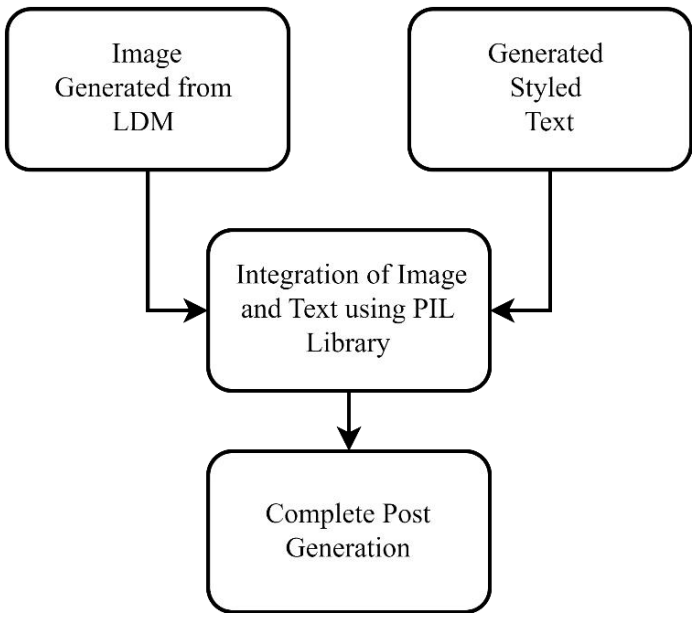


Figure 4-12: Integration of Generated Image and Styled Text

4.7 Flowchart of the Proposed System

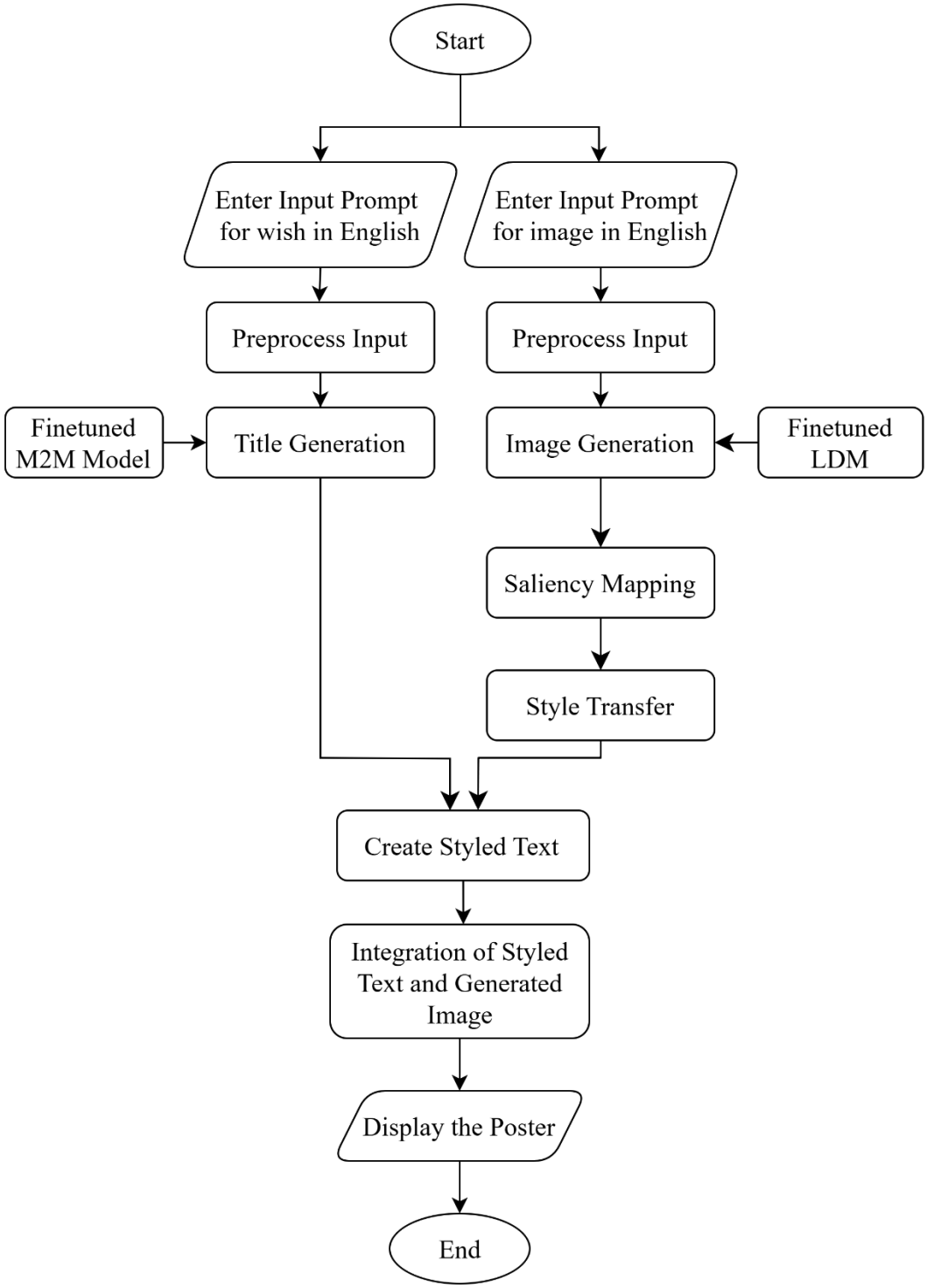


Figure 4-13: System Flowchart of Prompt to Poster Generation Model

Explain in 1-2 para. refencing Figure above

5. IMPLEMENTATION DETAILS

5.1 Text Dataset

The text dataset collected addresses different aspects such as structure, cleaning, and preprocessing that ensures quality and usability of the dataset. Additionally, it incorporated advanced prompt design and wish generation logic to create contextually relevant outputs.

5.1.1 Text Dataset Structure

The dataset is organized into two primary columns. The "Input" column contains prompts in English that specify the type of wish to be generated and the poster to be designed. The "Output" column holds the corresponding wishes in Nepali.

Example:

Input: Give me a poster for Chhath Puja 2077 with a sweet message.

Output: छठ पर्व २०७७ को शुभ अवसरमा सम्पूर्णमा सुख, शान्ति, समृद्धि, सुस्वास्थ्य, दीर्घायु एवम् उत्तरोत्तर प्रगतिको हार्दिक मंगलमय शुभकामना!

The dataset features wishes formatted into two distinct styles to enhance visual impact. The first style comprises brief, direct wishes, marked by an exclamation mark “!” to capture attention as main titles on posters.

Examples:

सूर्य उपासनाको महापर्व छठ पूजा २०७७ सालको दिव्य अवसरमा, तपाईंलाई र तपाईंको परिवारलाई हर्ष र समृद्धिको अनुपम शुभकामना!

२०८३ सालको छठ पर्वको शुभकामना!

The second style includes more elaborate and expressive festival greetings, ending with a purnabiram “।” These serve as subtitles, adding depth and warmth to the main titles.

Examples:

साथीहरूलाई मित्रताको उज्यालो तिहार २०८४ को शुभकामना! यो दीपावलीले तपाईंको दिनहरूलाई सधैँ सुखद र उज्यालो बनाओस्।

नयाँ वर्ष २०८१ को पावन अवसरमा हार्दिक शुभकामना! यो सालको नयाँ वर्षले तपाईंको जीवनमा नयाँ आशा, उत्साह, र प्रगति ल्याओस्।

Main Title: नयाँ वर्ष २०८१ को पावन अवसरमा हार्दिक शुभकामना! Subtitle: यो सालको नयाँ वर्षले तपाईंको जीवनमा नयाँ आशा, उत्साह, र प्रगति ल्याओस्।

Text

5.1.2 Data Cleaning and Preprocessing

To ensure the quality and accuracy of the dataset, several data cleaning and preprocessing steps were implemented. Spelling errors were corrected using dictionary resources and online spelling checker tools.

Raw Data:

दीपावली २०८० को अवसरमा हाम्रा सम्माननीय साझेदारलाई हार्दिक शुभकानना! यस चाडले तपाइलाई सुख, स्वास्थ्य र समृद्धिको आशिर्वाद प्रदान गरून्।

Spelling Corrected Data:

दीपावली २०८० को अवसरमा हाम्रा सम्माननीय साझेदारहरुलाई हार्दिक शुभकामना! यस चाडले तपाईंलाई सुख, स्वास्थ्य र समृद्धिको आशिर्वाद प्रदान गरून्।

Additionally, the dataset includes prompts without specific dates, paired with wishes that include a date. This approach ensures that every wish remains timely and relevant, and it also makes the wish generation process more consistent.

Example:

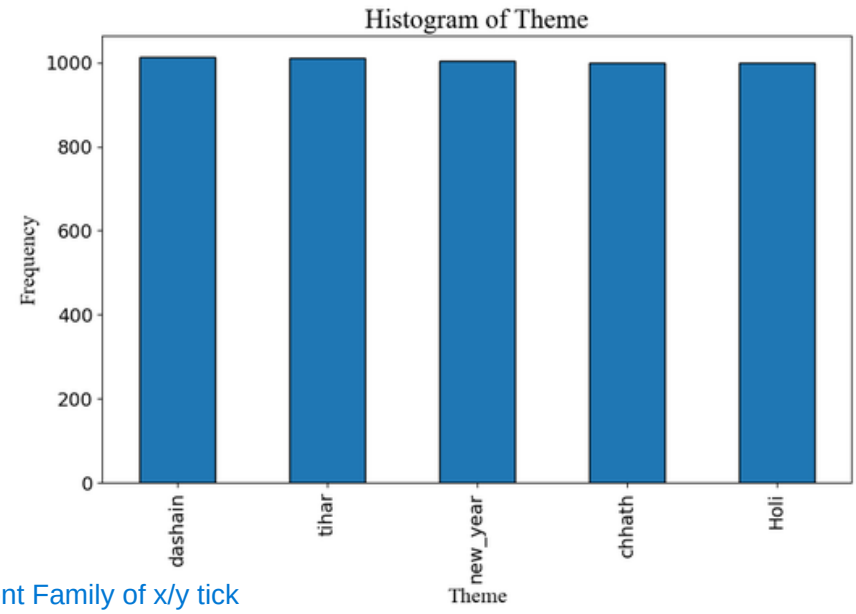
Input: Design a poster wishing everyone a Happy Holi.

Output: सबैलाई होली २०५२ सालको शुभकामना! तपाईंको जीवन रंगीन र खुशीयाली बनाओस्।

Text

5.1.3 Dataset Analysis

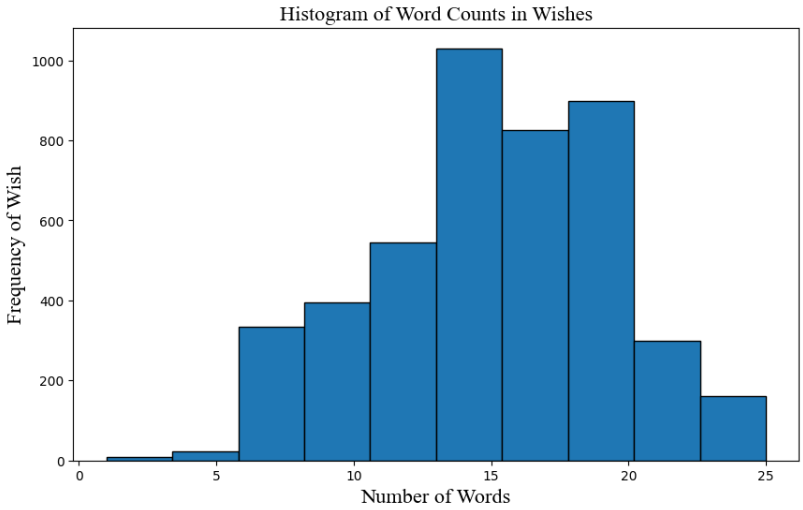
The dataset exhibits a balanced distribution of themes and a tendency towards medium-length wishes. This balance ensures that any analysis or modeling performed on this dataset would not be biased towards a particular theme or word count, allowing for more generalized and applicable insights.



Font Family of x/y tick
Overall font size/DPI

Figure 5-1: Histogram of Theme

Figure displays the frequency of different themes in the wishes, including Chhath, Dashain, Tihar, Holi, and New Year. Chhath appears most frequently, with 562 instances, while the other themes have relatively similar frequencies, around 500 instances. Dashain has 511 instances, Tihar has 510 instances, New Year has 501 instances, and Holi has 502 instances.



Same here

Figure 5-2: Histogram of Word Counts in Wishes

The length of the Nepali wishes was limited to 25 words to maintain consistency and readability across the dataset. Figure reveals that the word counts in wishes vary from 5 to 25 words, with a peak frequency around 10 to 15 words, indicating a preference for medium-length wishes. The distribution shows a bell-shaped pattern, suggesting that most wishes are neither too short nor too long.

These steps collectively contributed to a cleaner and more standardized dataset, enhancing the overall reliability of the generated wishes.

5.1.4 Prompt and Wish Generation Logic

For prompts and wishes generation, certain criteria were defined to make it easier for collecting relevant prompts and wishes.

Prompt Design

Prompts are collected in English and are crucial for generating titles and capturing user needs for image creation. They are designed to be clear and detailed, specifying the festival and, if necessary, the year.

Examples:

- Make a festive post for Dashain 2073.

- I need a concise and heartfelt Chhath Puja greeting designed for sharing with friends and relatives.
- Create a vibrant festive poster for Deepawali that conveys warm wishes for good health and prosperity to my family.

Each prompt should clearly specify the type of content needed and any relevant details about the festival to ensure that the resulting images effectively capture the intended message and spirit.

Wish Generation

The wish is generated in the Nepali language, based on relevant festivals. The wishes should be in a format suitable for titles in a poster, containing one sentence ending with an exclamation "!" The maximum length of wishes is 25 words, conveying blessings, wishes, and positive messages. A uniform structure of the dataset is created for easy analysis and training.

Example:

Input: Design a poster with wishes for New Year 2045 in short.

Output: नयाँ वर्ष २०२६ को पावन अवसरमा हार्दिक शुभकामना!

Prompts requesting short wishes result in brief and concise messages. Conversely, prompts asking for longer wishes yield more elaborate messages.

Examples:

Input: Make a Holi 2085 poster with a detailed message.

Output: वसन्त ऋतुको रङमय उत्सव, फागु पूर्णिमा २०८५ को शुभकामना! परिवार र साथीहरूसँगको बन्धनलाई बलियो बनाउँदै, रंगहरूको उत्सवलाई हर्षित र प्रेमपूर्ण बनाऔँ।

Generated wishes are tailored to suit the contexts of the prompt, ensuring they are relevant and personalized.

Contextual Examples:

Input: Compose a Chhath Puja greeting card mentioning the blessings of Surya Dev.

Output: छठ पर्व २०८३ को शुभकामना! सूर्यको ज्योतिले तपाईंको जीवन पथलाई उज्यालो बनाउँदै जाओस्।

Input: Poster to wish every family a blessed and prosperous Dashain 2087.

Output: सम्पुर्ण परिवारमा बडा दशैं २०५५ को हार्दिक मंगलमय शुभकामना! तपाईंको पर्व खुशी र सफलताले भरिपूर्ण होस्।

Input: Design a Deepawali poster expressing gratitude to your valued customers.

Output: हाम्रा आदरणीय ग्राहकहरूलाई सुखद तिहार २०८५ को शुभकामना! यो तिहारले तपाईंलाई निरन्तर सफलता र खुशी प्रदान गरिरहोस्।

This contextual sensitivity ensures that the generated wishes are relevant and personalized, meeting the specific requirements of the prompts.

5.2 Title Generation Pipeline

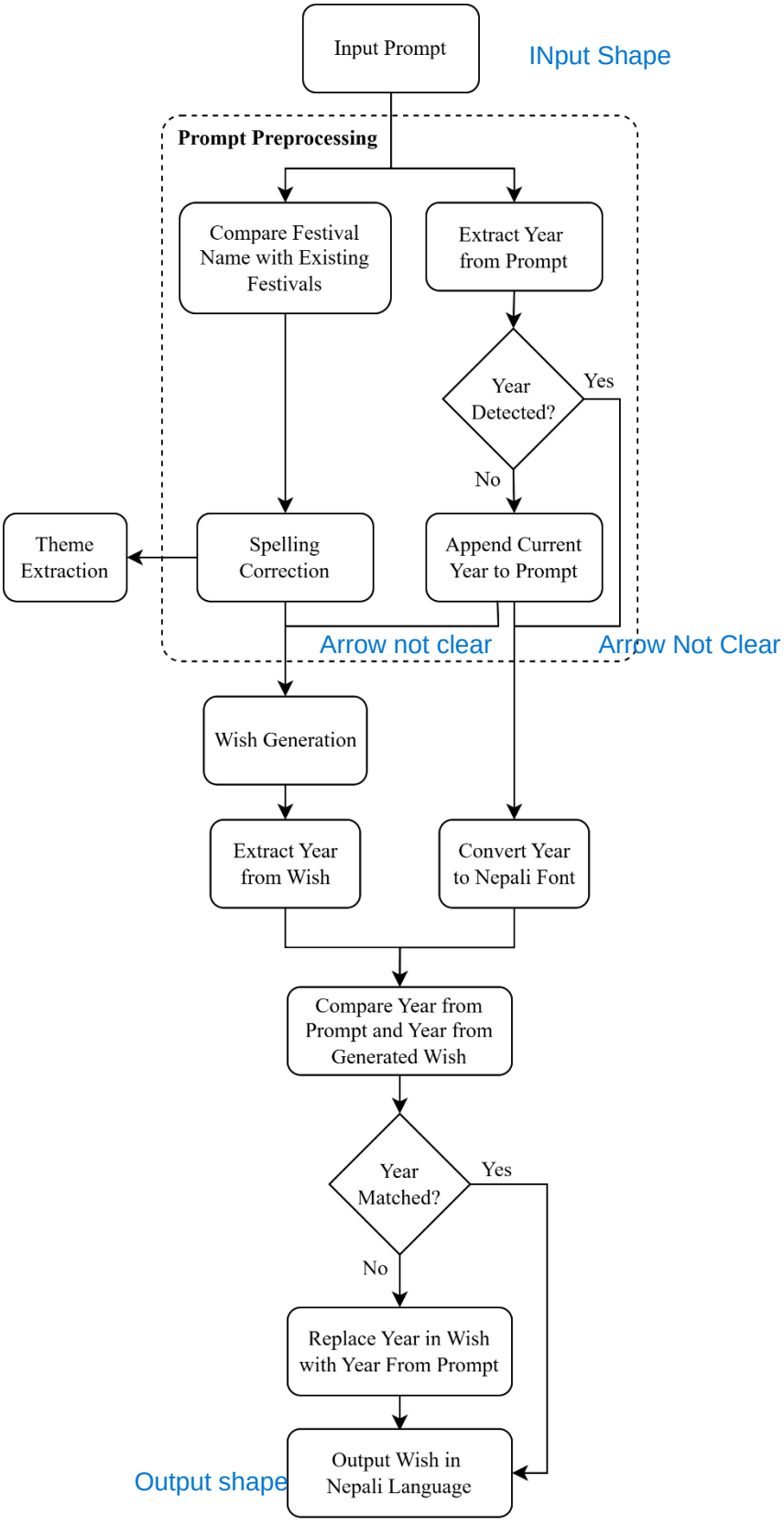


Figure 5-3: Flowchart of Title Generation Pipeline

Initially, the model receives a prompt in English from the user, which is then preprocessed.

The first step involves comparing the festival name in the prompt with the five festivals recognized by the model: Dashain, Tihar, Chhath, New Year, and Holi. If the similarity exceeds a predefined threshold, the prompt is identified as belonging to that festival. This identification is used for theme extraction as well as wish generation. In case the similarity is less than the threshold value, the user is asked to input a prompt again.

Next, the model extracts any year mentioned in the English prompt. If the model fails to detect year, the current year is appended to the festival name to form a complete prompt. The detected year, if present, or the appended year, in case of absence of year, is converted into Nepali font for future comparisons, while the complete prompt is passed to the wish generation module.

Once preprocessed, the prompt is used to generate a festival wish in Nepali language.

Again, the generated wish undergoes year extraction, and the extracted year is compared with the year from the prompt. If both years match, the wish is presented to the user as the final output. If the years are found to be different, the year in the generated wish is replaced with the year from the prompt, and the corrected wish is then presented to the user.

5.3 Hyperparameters for Finetuning M2M-100 Model

Table 5-1: Finetuned Hyperparameters for Finetuning M2M-100 Model

Parameter	**Value**
overwrite_output_dir	True
do_train	True
do_eval	True
eval_strategy	steps
logging_strategy	steps
save_steps	200
eval_steps	200

logging_first_step	True
logging_nan_inf_filter	True
save_strategy	steps
learning_rate	8e-5
per_device_train_batch_size	8
per_device_eval_batch_size	8
weight_decay	0.1
save_total_limit	2
logging_steps	100
max_steps	5000
gradient_accumulation_steps	1
lr_scheduler_type	linear
warmup_steps	600
warmup_steps	1000
restore_callback_states_from_checkpoint	True
report_to	wandb
predict_with_generate	True
dataloader_drop_last	True
load_best_model_at_end	True
metric_for_best_model	eval_loss
ignore_data_skips	False
greater_is_better	False
fp16	True

Table 5-2: Adam Optimizer Parameters for Finetuning M2M-100 Model

Parameters	**Value**
Beta1	0.9
Beta2	0.999
L2 Regularization	0.01
Gradient Clipping	1.0
Epsilon (ep)	0.00000001
Initial learning rate	0.00005

Total Training steps	10,000
Warm-up Steps	300

5.4 Image Dataset

The image dataset collected is organized, cleaned, preprocessed and analyzed to maintain high quality and consistency. Additionally, the incorporation of image generation logic ensures generation of context-aware visual outputs.

Image

5.4.1 Dataset Structure

The dataset for image generation is structured into two columns. The first column contains the image file names in .jpg format, while the second column holds the corresponding textual prompts for each image. For example, a row in the dataset may appear as follows:

First Column: photo1.jpg

Second Column: “An image for Dashain festival with kites and swings.”

This structure ensures a clear association between each image and its descriptive prompt, which serves as the foundation for model training and evaluation. The simplicity of the two-column format aids in efficient data parsing and processing during the pre-training and fine-tuning phases of the project.

Image

5.4.2 Data Cleaning

The images collected from various sources often contain undesirable elements such as embedded texts, watermarks, and noise, which can hinder the model’s ability to learn relevant features. To address this, the following cleaning operations were performed:

Text Removal

Any existing text embedded in the images was removed using manual and automated tools to prevent interference with visual feature extraction.

Watermark Elimination

How? explain more

Watermarks were detected and removed to ensure the dataset contained clean visuals.

Format Standardization

All images were converted to a consistent format and resolution of 512x512 suitable for training.

How explain more

These cleaning procedures ensured that the dataset was free from distractions and maintained a high standard of quality for model training.

5.4.3 Data Preprocessing

Data preprocessing is an essential step to optimize the dataset for training by enhancing image quality and augmenting the dataset to improve model generalization. The preprocessing tasks were carried out in two stages: image enhancement and data augmentation.

Image Enhancement

How explain more

To improve the quality of the images, an online application was utilized. This application ensured that the images met the required quality standards for training purposes. Low-resolution images were upscaled and blurry images were sharpened to highlight important features.

Image Augmentation

Include mathematics

Image augmentation techniques were applied to enhance the diversity and robustness of the dataset. Several geometric transformations were applied to change the spatial properties of the images. Horizontal flips were performed to create mirrored versions of the images, random rotations were done to apply angular shifts. Some images were rotated in 90-degree increments like 90°, 180°, or 270°. Translations were used to shift images horizontally and vertically. Scaling transformations were applied to adjust the image size, while maintaining proportions. Furthermore, random shear was employed to distort images along the horizontal or vertical axes to create slanted perspectives.

Why extra space?

Color adjustments were made to imitate various lighting scenarios and visual impacts. Saturation levels were changed to either highlight or de-emphasis colors, hue modifications were done to give warm or cold tones, contrast levels were changed to improve the separation between light and dark regions, and brightness modifications were made to change the photos' overall lightness or darkness. CLAHE was used to enhance local contrast, especially in dim areas, and random gamma adjustments were added to dynamically modify brightness and contrast.

Furthermore, the images were manipulated by adding random noise to simulate real-world imperfections, and blurring was also applied. Sharpness modifications were made to enhance and soften the images. Random cropping was used to remove sections of the images. Random erasing was performed to obscure specific areas so that the model would work on missing data. Compression was also applied at different levels to simulate different storage and transmission conditions.

These augmentation techniques were carefully applied to ensure the model was exposed to a wide range of scenarios, improving its robustness and adaptability in real-world applications.

Prompt Augmentation

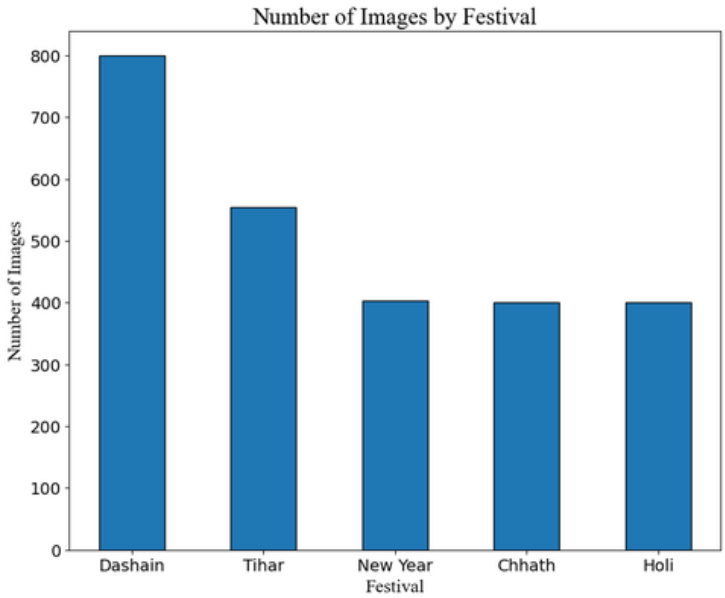
In prompt augmentation, the textual prompts associated with images were modified to reflect the transformations applied to the images to ensure that the images and their descriptions align well. Various functions were implemented to achieve prompt augmentation.Space? Words such as “image,” “illustration,” or “photo” in the prompts were replaced with more specific terms like “warm image,” “cold image,” “saturated image,” “dark image,” or “bright image” to reflect the different transformations of the images. These terms indicated the application of warm hues, cold hues, enhanced color intensity, reduced brightness, and increased brightness, respectively.

Similarly, the prompts were augmented for other positional manipulations, color adjustments, and image distortions. This ensured that each prompt accurately described the visual changes made to the images, improving the coherence of the dataset.

Image

5.4.4 Dataset Analysis

The dataset consists of a balanced distribution of images and their corresponding prompts. This balance ensures that analyses or models developed using this dataset remain unbiased toward specific features or word counts, leading to insights that are more generalizable and widely applicable.



xtick,ytick font fam?, overall font size?

Figure 5-4 Histogram of Image Dataset Distribution

The histogram illustrates the frequency of images in the dataset, focusing on festivals such as Chhath, Dashain, Tihar, Holi, and New Year. The dataset includes approximately 800 images for Dashain, 550 for Tihar, and around 400 images each for Chhath, Holi, and New Year.

The prompts used for image generation are concise but do not have a strict word limit. The efforts to clean and standardize the dataset have improved its overall quality, ensuring reliable image generation based on the provided prompts.

The length of the prompt for image generation is short but there is no word limit.

These steps collectively contributed to a cleaner and more standardized dataset, enhancing the overall reliability of the generated images based on prompt.

5.4.5 Prompt and Image Generation Logic

For prompts and images required for image generation, certain criteria were defined to make it easier for collecting relevant prompts and wishes.

Prompt Design

Prompts are written in English and play a crucial role in capturing the user's requirements for image creation. They are designed to be clear, detailed, and specific to the festival being represented. The prompts and images mainly focus on the festival's unique assets, culture, and foods or items specifically associated with that festival.

Examples:

Left margin make consistent with other bullets

• Image of two friends dancing joyfully during Holi, covered in colorful powders with plates of colors around them.

• Tihar image of Goddess Laxmi sitting on a lotus flower, people dancing and tihar lights.

Each prompt explicitly outlines the desired content and festival-related details to ensure the generated images accurately reflect the intended message and festive spirit.

Image Generation

Images are generated based on the user's prompts, adhering to a standard resolution of 512 x 512 pixels. These images are designed to convey the relevant information about the specified festivals. The dataset is structured uniformly, facilitating seamless analysis and model training.

Example:

Input: Image of children lighting firecrackers in a house decorated with flower garlands, lights, diyos and rangolis during Tihar.

Output:

Figure 5-5: Example 1 of Image Generation

Input: Holi image with a boy and a girl playing around with colors and waterguns and color splashed in the background.

Output:

Figure 5-6: Example 2 of Image Generation

5.5 Hyperparameters for Finetuning LDM model

Table 5-3: Finetuned Hyperparameters for LDM model

Parameter	**Value**
batch_size	1
learning_rate	0.00005
num_epochs	10
num_inference_steps	1000
num_workers	2
weight_decay	1e-3
num_training_steps	500
num_validation_steps	100
save_every_n_steps	100
image_steps	10
patience	3
onecycle_max_lr	5e-2
accumulation_steps	4

5.6 Saliency Mapping

Saliency mapping is a technique used to identify the most prominent and informative regions of an image, with areas that significantly influence a model's predictions being highlighted. A visual explanation of how models prioritize different areas of an image is provided.

Mathematically, saliency maps are derived by computing the gradient of the class score concerning the input image.Given a class score Sc(I) for an image I, the saliency map M is defined as:

$$M(x, y) = \frac{Sc(I)}{I(x,y)} \quad 5.1$$

where,

Sc(I) is the score of the target class ccc produced by the model for the input image I.

I(x,y) represents the pixel value of the input image I at position (x,y)

Sc(I)/I(x,y)is the gradient of the class score with respect to the pixel value, which quantifies how sensitive the score is to changes in that pixel.

In our project, saliency mapping is employed to identify empty spaces in images generated by a Latent Diffusion Model (LDM). The process involves several key steps:

Preprocessing

Input images are resized and normalized to ensure compatibility with the pre-trained DenseNet-201 model.

Gradient-based Saliency Maps

The gradient of the model's output with respect to the input image is computed, highlighting the pixels that contribute most to the model's prediction.

Normalization and Resizing

The raw saliency map is normalized and resized to match the original image dimensions.

Contour Detection

Computer vision techniques, such as contour detection, are applied to the processed saliency map to identify low-saliency regions suitable for text placement.

Region Selection

A threshold is applied to isolate these regions, and the region with the largest bounding rectangle is selected as the ideal location for text overlay.

Visualization

The saliency map and the selected region are visualized to ensure that the textual overlay does not obstruct critical areas of the image.

By using saliency mapping, it is ensured that festival-related visual elements, such as cultural symbols and decorations, remain unobstructed, enhancing the aesthetic and cultural relevance of the generated posters.

5.7 Text Styling

The text styling algorithm transforms the text generated by the M2M model into an aesthetically pleasing styled text by determining the text's colour, font style and size.

5.7.1 Extraction of Text Colours

The dominant colour along with a palette of six other colours are extracted from the image using the ColorThief library. The color contrast of each colour with the dominant colour is calculated using RGB distance calculations. The colour with the highest colour contrast is selected as the text colour. However, if the highest colour contrast doesn't provide enough readability, determined by a threshold, white or black colours are assigned as text colour based on the luminance of the dominant colour.

Original Image:

Figure 5-7 Original Image for Text Styling

Extraction of Dominant Colour:

Figure 5-8 Dominant Colour of the Original Image

Extraction of Colour Palette:

Figure 5-9 Colour Palette of the Original Image

Determination of Text Colour based on Dominant Colour:

Figure 5-10 Dominant Colour and Selected of Text Colour for Text Styling

5.7.2 Selection of Font Style

From a predefined collection of 51 Nepali-compatible font families, which includes a total of 371 font variations, a font style is selected randomly to style the text.

5.7.3 Selec...

5.7.3 Selection of Font Size

A font size is randomly selected from a specified range. If the wish contains two sentences, two separate font sizes are selected, the larger one for the wish and the smaller one for the subtitle while only one font size is selected for one sentence wishes.

5.7.4 Text Placement

The text is styled using the selected colour, font style and size. The styled text is warped to ensure that the text fits within the given width. After which the styled text is converted into image form and placed on the image.

6. RESULTS AND ANALYSIS

6.1 Result on Training M2M Model

After training on the 2586 dataset of five festivals, we achieved the following results:

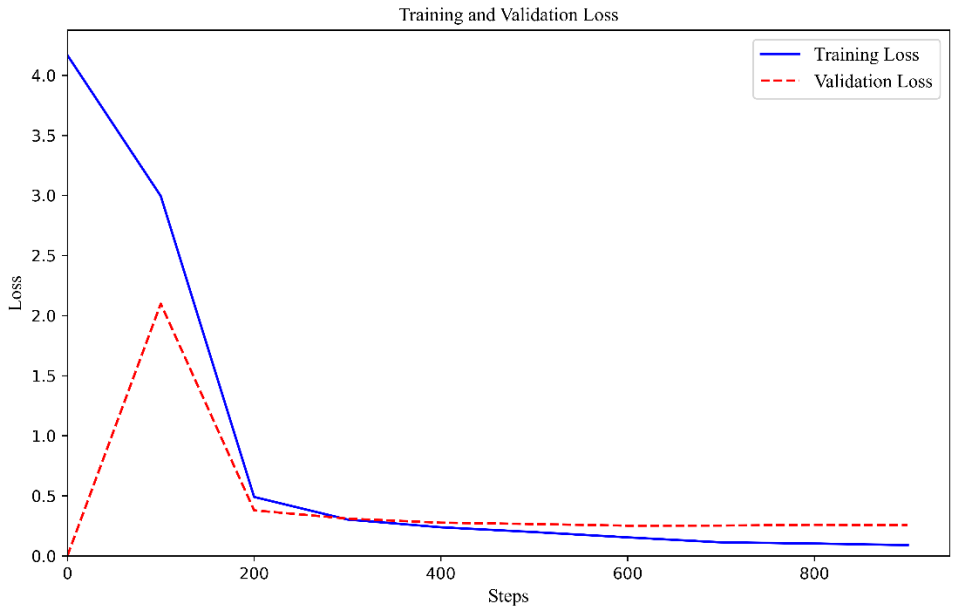


Figure 6-1: Training Loss for Text Dataset

The training and validation loss plot illustrates the performance of the model during training and validation phases. Initially, the training loss was observed to be 2.994800. After 200 steps, the loss decreased drastically to 0.490500. Subsequently, the model experienced slight decrease in loss every 200 steps resulting in the training loss value of 0.089600 at 1800 steps. Similarly, the validation loss was initially observed to be 2.098300. It significant decreased to 0.378988 in the interval of 200 steps after which the rate of decrease of validation loss was minimal, resulting in the final validation loss of 0.255655 at the end of 1800 steps.

precision figure in numbers max 3-4 digits

Mention, refer each figure in required places

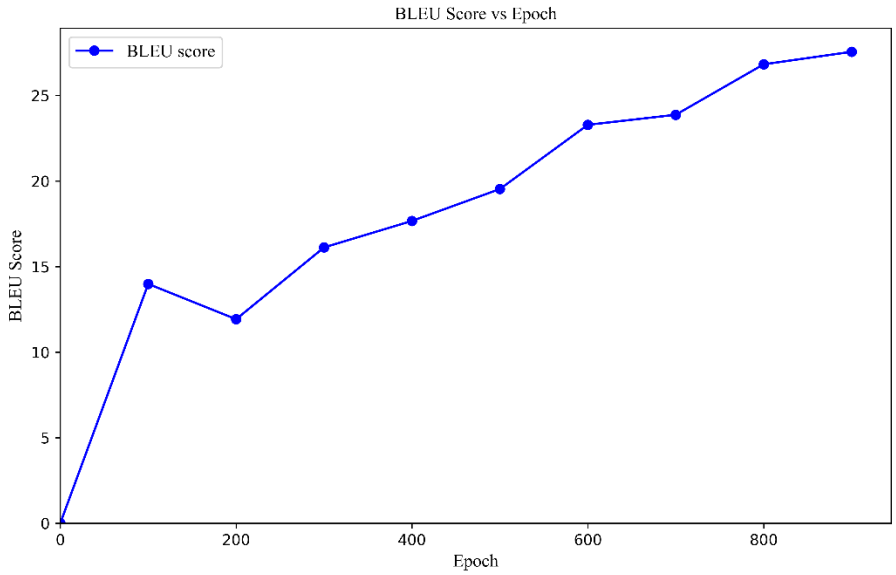


Figure 6-2: BLEU Score Evaluation for Text Dataset

From the BLEU score vs steps plot, it is observed that it started at 13.99 for the first 200 steps, dropped to 11.92 in the next 200 steps, but then consistently increased every 200 steps, reaching 27.55 by 1800 steps. This shows the model's performance improved over time, achieving a moderate similarity between generated and reference text by the end of training.

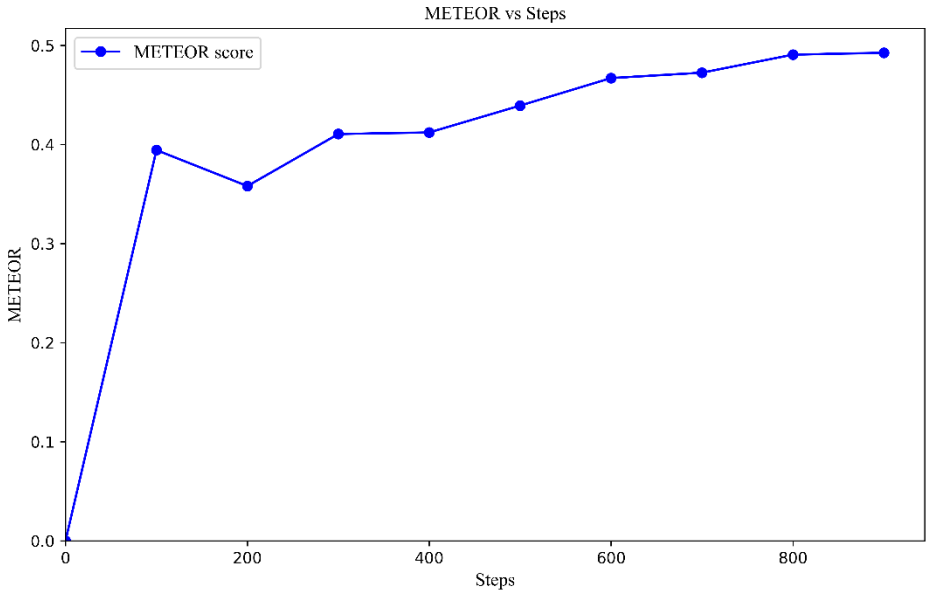


Figure 6-3 : METEOR for Text Dataset

The Meteor Score vs steps plot follows a similar pattern as BLEU score plot. Initially, the meteor score is 0.394296. The Meteor score slightly decreases to 0.358028 for the next 200 steps. However, the score improves steadily after that reaching 0.492510 by the end of 1800 steps which indicates reasonably good precision, recall and synonymy match in model's performance.

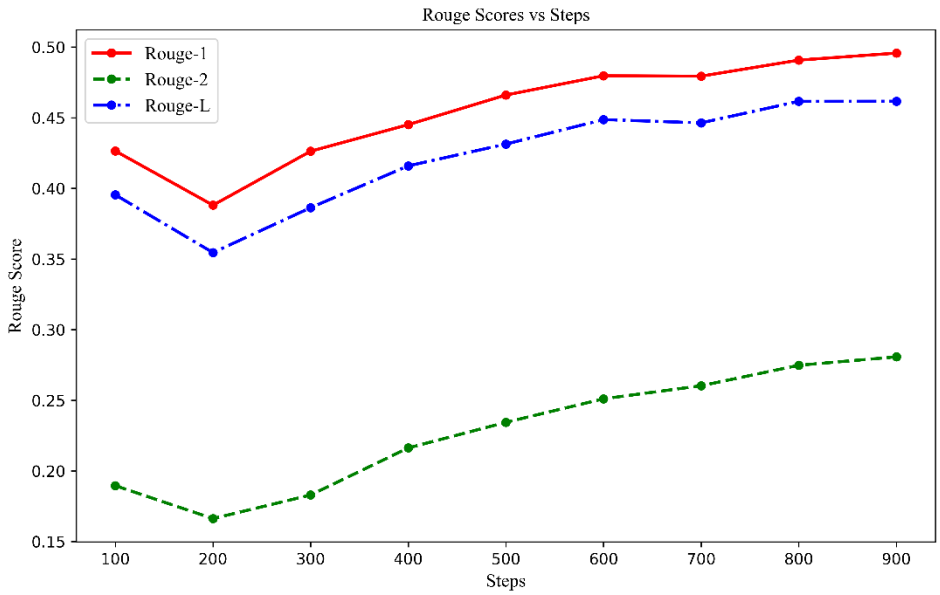


Figure 6-4: ROUGE Score for Text Dataset

The rouge vs steps plot illustrates the value of ROUGE-1, ROUGE-2 and ROUGE-l across different training steps. The initial value of ROUGE-1 is 0.426452, ROUGE-2 is 0.189550 and rouge-l is 0.395388. All three curves follow a similar pattern of slight decrease from 200 to 400 steps followed by a steady increase after that resulting in the final values of ROUGE-1, ROUGE-2 and ROUGE-l as 0.495723, 0.280740 and 0.461637 respectively at the end of 1800 steps reflecting an overall improvement in content overlap. Since ROUGE-1 has the highest value, the model is most effective at capturing individual word matches while it struggles in preserving word pairs and sequences as shown by ROUGE-2. Also, the value of rouge-l reflects a balanced performance in maintaining longer, coherent sequences.

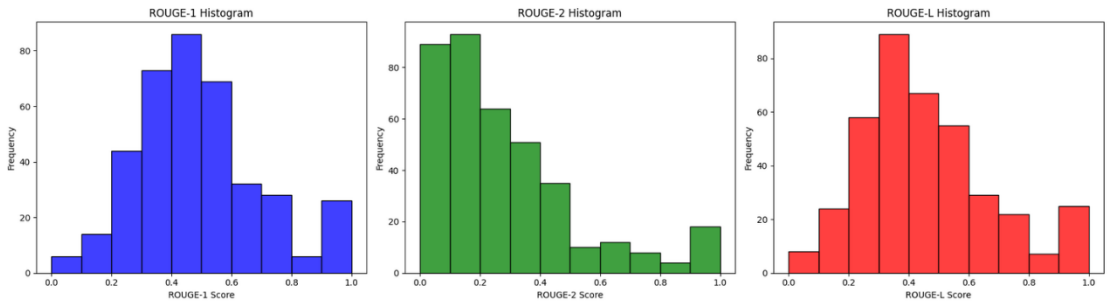


Figure 6-5: Histograms of ROUGE Scores

In each plot, the x-axis represents the F1 scores, while the y-axis indicates the frequency of each score. These plots illustrate the frequency distribution of F1 scores across the dataset. The histogram for ROUGE-1 reveals a near-symmetric distribution, suggesting that the majority of samples achieve scores between 0.2 and 0.8. This indicates a relatively balanced overlap between the generated and reference text at the unigram level. Similarly, the ROUGE-L histogram also shows an almost symmetric distribution, with most scores ranging between 0.2 and 0.8. This reflects a balanced overlap at the l-gram level, capturing longer sequences of matching text. In contrast, the ROUGE-2 histogram displays a right-skewed distribution, indicating that the majority of samples achieve scores up to 0.4. This skewness suggests that the overlap between the generated and reference text at the bigram level is generally lower compared to unigrams and l-grams.

Overall, these histograms provide a comprehensive view of the frequency and distribution of F1 scores across the validation dataset, highlighting the degree of overlap between the generated and reference texts for different n-gram levels.

We loaded our pre-trained model, which is capable of producing a wide range of wishes based on user inputs. The model takes into account various factors, such as the length and style specified by the user, to generate customized and diverse wish messages. The user can specify in the prompt the type of wish message they want on the poster, whether it be long, short, or any specific style they prefer. Additionally, if the user does not specify the year in the prompt, the current year will be added to the wish message.

Here are a few examples of the output from our trained model:

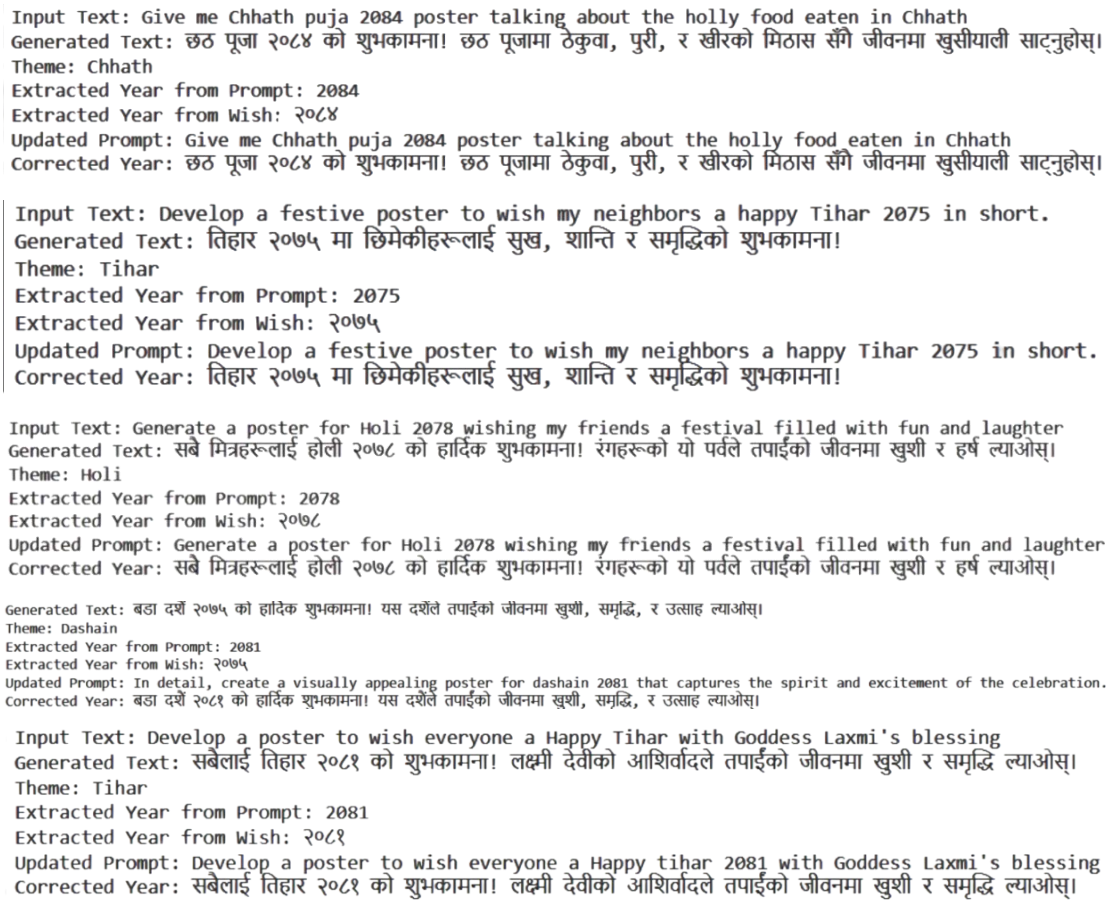
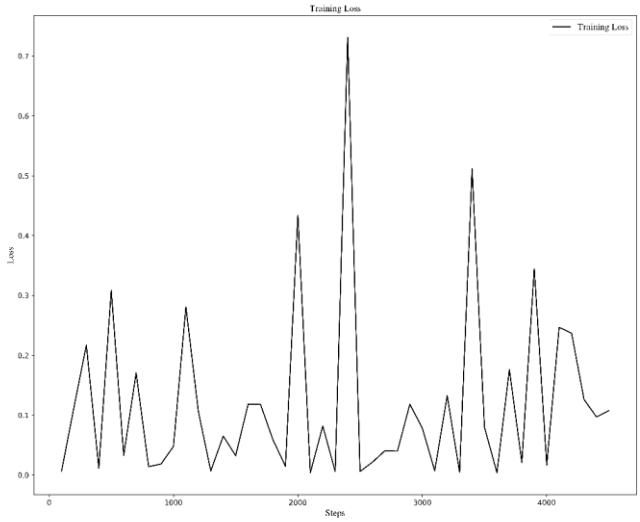


Figure 6-6: Screenshot of Input Text given to the Model and the Generated Output Text from the Model

6.2 Result on Training LDM Model



Labels, legends, title increase size

Figure 6-7: Training Loss vs Steps for Image Dataset

The training loss shows significant fluctuations, likely due to the small batch size of 1 causing high gradient variance and a conservative learning rate of 0.00005, which may slowly convergence.

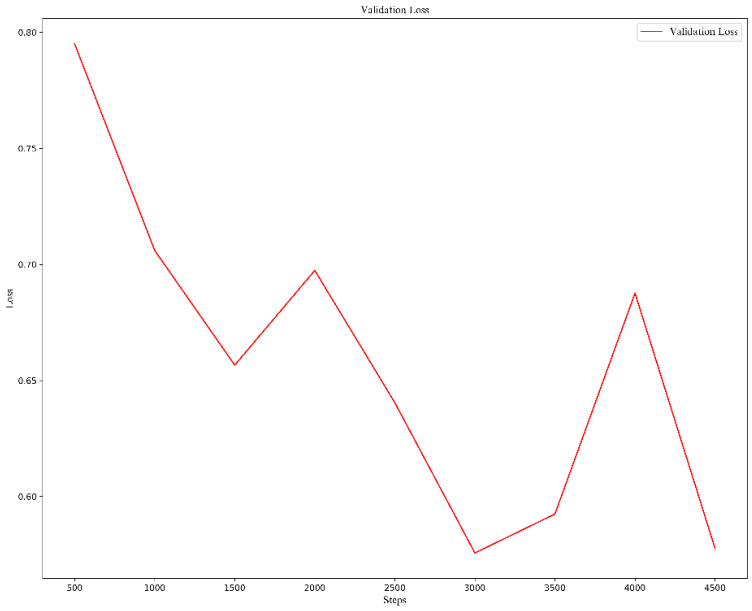


Figure 6-8: Validation Loss vs Steps for Image Dataset

Validation loss shows minor improvements, suggesting some benefits from the optimizer and cosine scheduler, but over-regularization from the weight decay 1e−3 and limited inference steps (1000) may hinder generalization.

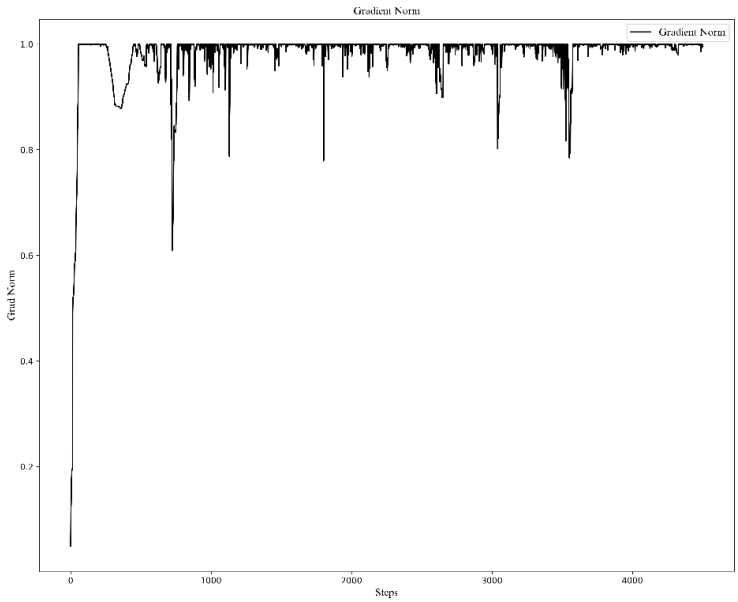


Figure 6-9: Gradient Normal vs Steps for Image Dataset

The flat weight norm indicates constrained weight updates due to strong regularization and learning rate settings, potentially limiting the model's adaptability. To address these issues, we will increase the batch size, explore a slightly higher learning rate with more aggressive scheduling, adjust weight decay to allow dynamic weight evolution, and extend inference steps to capture quality improvements better.

The gradual decrease in loss with occasional spikes suggests that the model is improving over time, though fluctuations may indicate issues like small batch sizes or high learning rates. Increasing the batch size and fine-tuning the learning rate could help stabilize the loss curve.

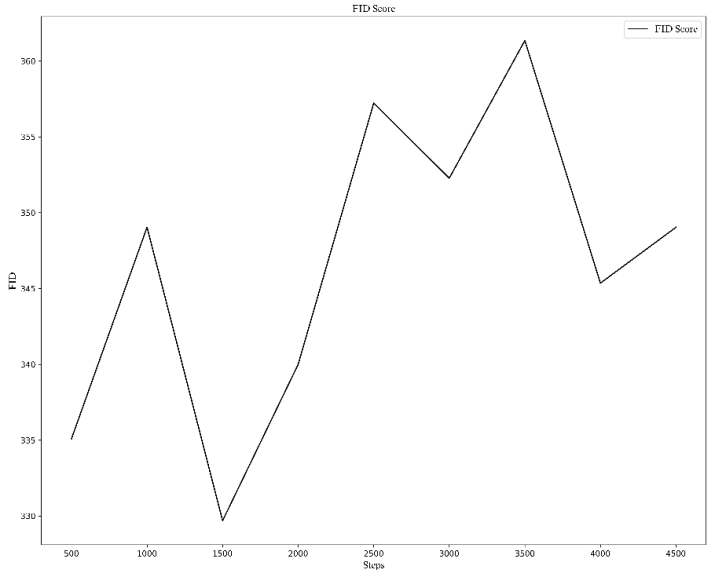


Figure 6-10: FID Score vs Steps for Image Dataset

The fluctuating FID score indicates inconsistent image quality, likely due to insufficient training steps or a suboptimal learning rate. Increasing the training duration and adjusting the learning rate could lead to more stable improvements in image quality.

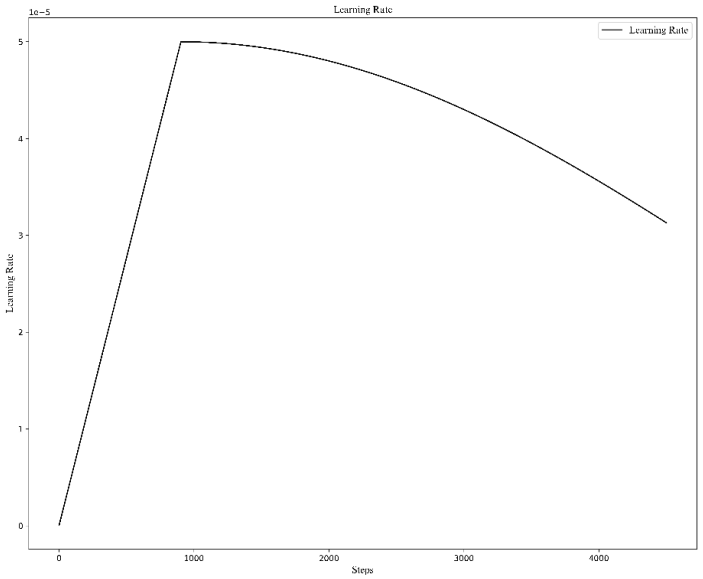


Figure 6-11: Learning Rate vs Steps for Image Dataset

The cosine decay of the learning rate shows the expected pattern, which should help the model converge. Fine-tuning the learning rate schedule, possibly by increasing warm-up steps, could better align with the dataset and training progress.

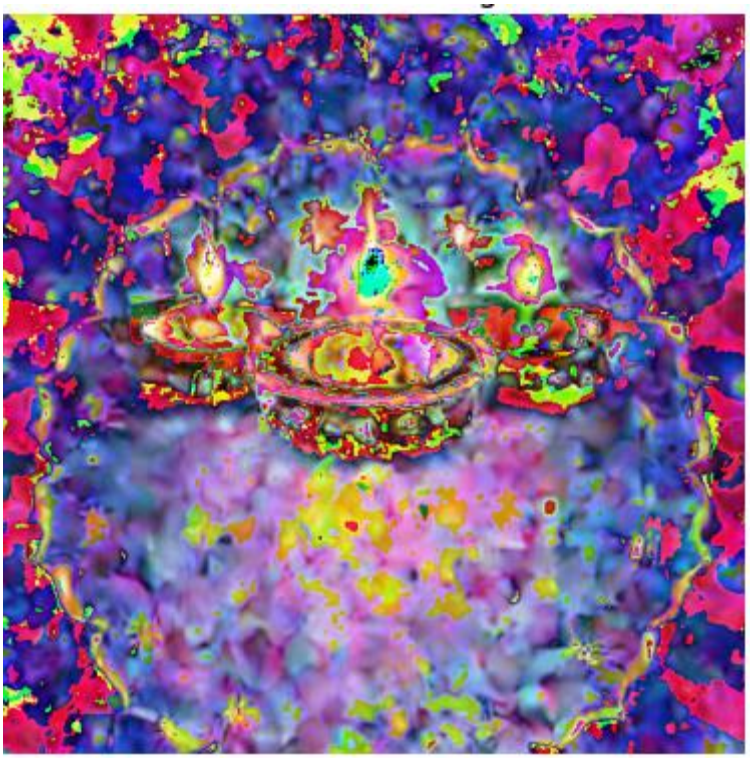


Figure 6-12: Example 1 of Image Generated using LDM

This image, featuring a Diyo, captures the essence of the Tihar festival. However, the lack of clarity suggests that the model requires further analysis and refinement to generate sharper, more detailed images in the future.

Figure 6-13: Example 2 of Image Generated using LDM

This image, depicting the Chhath festival with a woman worshipping the Sun, captures the essence of the celebration. However, the features representing the woman are not entirely clear.

Figure 6-14: Example 3 of Image Generated using LDM

This image, representing Goddess Durga for the Dashain festival, is the best generated so far during model validation. While it captures the essence of the celebration, the image appears somewhat blurred. To enhance the quality of the generated images, we plan to train a VQ-VAE (Vector Quantized Variational Autoencoder) model, which will map the images into a latent space. This approach should help refine the image generation process by preserving finer details and reducing blurriness, ensuring that future generations are sharper and more visually appealing.

6.3 Result of Saliency Mapping

After the image generation, saliency mapping is applied to identify empty spaces in the generated image to place text appropriately. The output shows the original image, saliency map, and an image with a rectangle to visualize empty spaces in the original image.

The original image on the left showcases a vibrant design with colorful elements. The saliency map in the center highlights areas of high visual attention, with brighter regions

indicating less focus or empty spaces. On the right, a red rectangle outlines the empty space in the original image, providing a designated area for adding text or festival wishes without interfering with the primary visual elements.

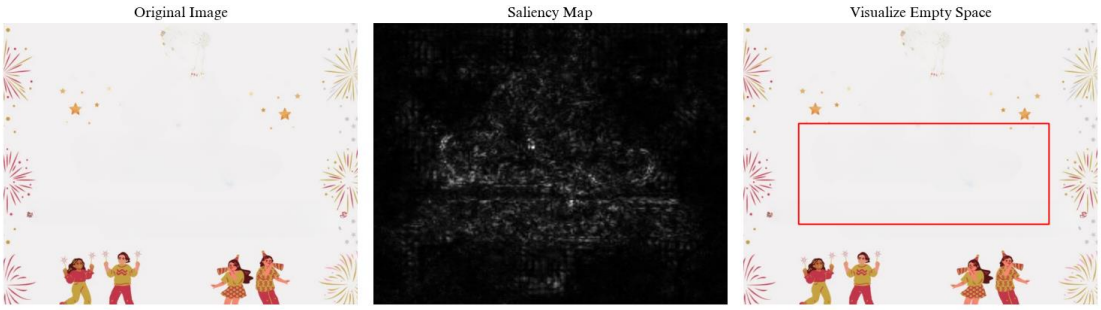


Figure 6-15: Example 1 of Saliency Mapping

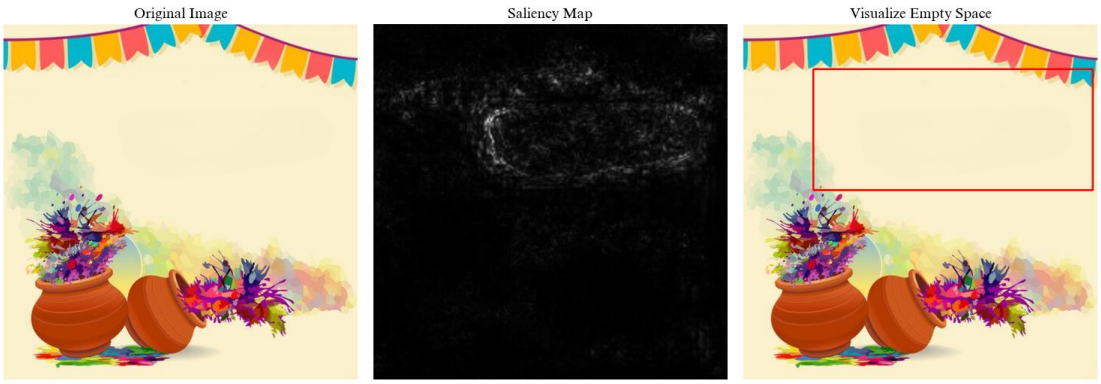


Figure 6-16: Example 2 of Saliency Mapping

6.4 Result of Text Styling

The wish in Nepali language is styled by selecting a text colour having enough contrast with the background along with suitable font style and size. The styled text is then manually positioned on the image generated by the LDM, resulting in a vibrant and visually appealing poster.

Figure 6-17: Example 1 of Text Styling

Figure 6-18: Example 2 of Text Styling

7. REMAINING TASK

7.1 Fine-tuning LDM for Image generation

We will fine-tune the LDM model to generate images that precisely reflect the descriptions provided. For instance, a Dashain poster will include scenes like people putting tika, flying kites, or playing on swings, along with Dashain wishes. For Tihar, the model will create posters featuring decorative lights, rangoli designs, and the worship of animals like dogs and cows.

To ensure the images are visually appealing and culturally relevant, each poster will align with the specific context and message of the description, featuring customs and events specific to each festival for a meaningful audience experience. For example:

For a input text prompt: “Craft a poster for Dashain incorporating festive elements and designs.”

The LDM model will generate an image that includes the features mentioned in the prompt.

Figure 7-1: Dashain Poster Generation using LDM

7.2 Integration

We will combine text styling algorithm and saliency mapping algorithm to place the styled text to optimally position the styled text on festive image generated using LDM to develop a visually appealing poster that delivers the intended message, making it suitable for various applications such as promotional and decorative purposes.

7.3 Creating a User-Friendly Interface

We will create a user-friendly interface that allows users to input text prompts easily and view the generated images. This interface will be intuitive, enabling users to quickly generate and download the desired posters with minimal effort.

8. APPENDICES

Appendix A: Project Budget

Table 8-1: Budget Estimation

Item	Price (NPR)
Printing	5000.00
Colab Resource (1400 per month)	4200.00
Domain (per year)	500.00
Hosting (per year)	2000.00
Grand Total	**11,700.00**

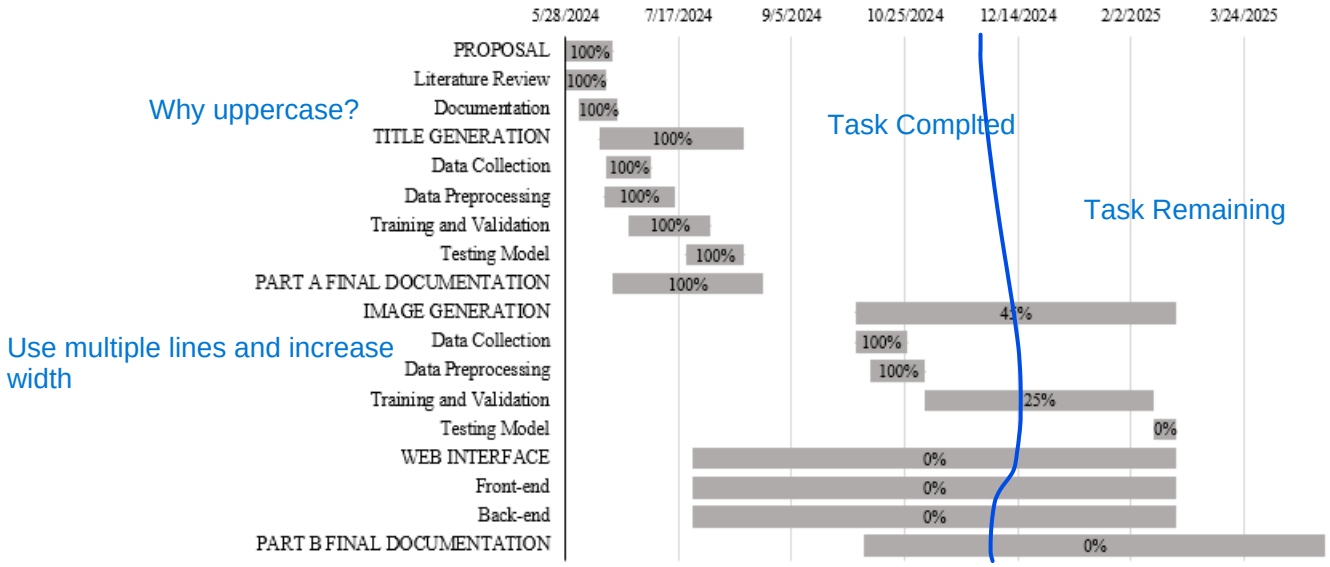
comma

comma

comma

Appendix B: Project Schedule

Figure 8-1: Project Timeline Gantt chart



Appendix C: Review of Base Paper I

Author(s)/Source: Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, Armand Joulin	
Title: Beyond English-Centric Multilingual Machine Translation	
Website: https://arxiv.org/abs/2010.11125	
Publication Date: 21st October, 2020	
Journal: Journal of Machine Learning Research	**Place:** New Orleans, LA, USA
Author's position/theoretical position: The authors are researchers at Facebook AI Research (FAIR), specializing in machine learning and multilingual machine translation.	
Keywords:BLEU points, Dense scaling, M2M-100	

Important points	**Page No.**
1. Implements extensive data filtering and mining from Common-Crawl, including language and length filtering to ensure high-quality datasets.	8
2. Uses a bridge language group mining strategy, grouping 100 languages into 14 groups and defining bridge languages for efficient data mining.	9-10
3. Employs a Transformer-based sequence-to-sequence architecture with increased capacity through dense scaling and language-specific sparse parameters.	18
4. M2M-100 outperforms English-centric models, showing an average improvement of over 10 BLEU points for non-English translations and achieving competitive results with bilingual models.	23-26

Essential Background Information: Many-to-Many multilingual translation model can translate directly between multiple languages unlike traditional English-centric machine translation.
Overall Argument or Hypothesis: The paper argues that a true Many-to-Many multilingual translation model can outperform English-centric models by directly translating between language pairs, improving translation quality for non-English languages.
Conclusion: The Many-to-Many model significantly improves translation quality for non-English languages and offers a robust solution for multilingual machine translation without relying on English.

Supporting Reasons: 1. M2M-100 model shows a gain of over 10 BLEU points in non-English translations compared to previous models, demonstrating improved accuracy and efficiency. 2. Developed and open-sourced a dataset with 7.5 billion training sentences, covering thousands of language directions, enhancing translation accuracy. 3. Utilizes dense scaling and language-specific sparse parameters to handle the complexities of multilingual translation, maintaining high performance. 4. Employed a mixture-of-experts strategy to optimize training, ensuring low-resource languages receive adequate attention, improving overall performance and generalization.
Strengths of the line of reasoning and supporting evidence: The paper provides strong evidence through comprehensive experiments and significant performance improvements over existing models. The open sourcing of the dataset and scripts enhances reproducibility and invites further research, supporting the robustness of the model.
Flaws and Weaknesses: One potential weakness is the need for massive computational resources, which may not be available to all researchers. The model performs well but its complexity could limit its use in resource-constrained settings.

Appendix D: Review of Base Paper II

Author(s)/Source: Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Björn Ommer	
Title: High-Resolution Image Synthesis with Latent Diffusion Models	
Website: https://arxiv.org/abs/2112.10752	
Code Link: https://github.com/CompVis/latent-diffusion	
Publication Date: 13 Apr 2022	**Access Date:** 27 September 2022
Journal: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2022	**Place:** New Orleans, LA, USA
Author's position/theoretical position: Four are students of Ludwig Maximilian University of Munich & IWR, Heidelberg University, Germany; one is from runwayml.	
Keywords: Image and video synthesis and generation	
Important points	**Page No.**
1. Perceptual compression model consists of an autoencoder trained	3

by combination of a perceptual loss and a patch-based adversarial objective.

2. Compared to the high-dimensional pixel space, latent space is more suitable for likelihood-based generative models. 4
3. We turn DMs into more flexible conditional image generators by augmenting their underlying UNet backbone with the cross-attention mechanism which is effective for learning attention-based models of various input modalities. 4

Essential Background Information: Training and optimizing Diffusion models directly in pixel space takes up huge computational resources.

Overall Argument or Hypothesis: Latent Diffusion Models (LDMs) reduce computational resources while maintaining quality and flexibility.

Conclusion: LDMs achieve state-of-the-art performance in image inpainting, class-conditional image synthesis, text-to-image synthesis, unconditional image generation, and super-resolution.

Supporting Reasons:

1. Encoder in autoencoder: Encodes images into 256x256 latent representations preserving perceptual details.
2. Time-conditional UNet: Streamlined architecture preserving small details, suitable for time-dependent data.
3. Cross-attention mechanism: Enhances image generation alignment with input conditions.
4. LDM with downsampling factors 4 and 8: Optimal for high-quality synthesis results.
5. Latent diffusion models are more suitable for likelihood-based generative models as latent space reduces high computational resources needed for the models.
6. KL-regularization and VQ-regularization are used to make the latent representation more reliable, structured and able to work with different kinds of data.

Strengths of the line of reasoning and supporting evidence:

1. LDM-4 achieves FID of 5.11 on CelebA-HQ, superior performance on the FFHQ dataset.
2. Class-conditioned LDM on ImageNet outperforms the state-of-the-art diffusion model.
3. LDM-4 sets a new FID score of 9.39 on image inpainting.

Flaws and Weaknesses:

1. Slower sequential sampling process compared to GANs.
2. Questionable precision in some high-precision applications.
3. Potential for data manipulation and misinformation with generative models like LDMs.

References

[1] A. Fan, S. Bhosale, H. Schwenk, Z. Ma, M. Baines, O. Celebi, G. Wenzek, V. Chaudhary, N. Goyal, T. Birch, V. Liptchinsky, S. Edunov , E. Grave, M. Auli and A. Joulin, "Beyond English-Centric Multilingual Machine Translation," 2020.

[2] J. Ho, A. Jain and P. Abeel, "Denoising Diffusion Probabilistic Models," 2020.

[3] B. Zhang, S. Gu, B. Zhang, J. Bao, D. Chen, F. Wen, Y. Wang and B. Guo, "StyleSwin: Transformer-based GAN for High-resolution Image Generation," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[4] S. Guo, Z. Jin, F. Sun, J. Li, Z. Li, N. Cao and Y. Shi, "Vinci: An Intelligent Graphic Design System for Generating," 2021.

[5] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu and M. Chen, "Hierarchical Text-Conditional Image Generation with CLIP Latents," 2022.

[6] R. Rombach , D. Lorenz , P. Esser, . B. Ommer and A. Blattmann, "High-Resolution Image Synthesis with Latent Diffusion Models," 2022.

[7] D. Huang, J. Li, C. Liu and J. Liu, "AUPOD: End-to-End Automatic Poster Design by Self-Supervision," *IEEE Access,* vol. 10, pp. 47348-47360, 2022.

[8] M. Tao, H. Tang, F. Wu, X. Y. Jing, B.-K. Bao and C. Xu, "DF-GAN: A Simple and Effective Baseline for Text-to-Image Synthesis," 2022.

[9] A. Sauer, T. Karras, S. Laine, A. Geiger and T. Aila, "StyleGAN-T: Unlocking the Power of GANs for Fast Large-Scale Text-to-Image Synthesis," 2023.

[10] J. Lin, M. Zhou, Y. Ma, Y. Gao, C. Fei, Y. Chen, Z. Yu and T. Ge, "AutoPoster: A Highly Automatic and Content-aware Design System for Advertising Poster Generation," 2023.

Not Enough citation, look for missing citations