Team 5 - Project F Final Update

Stable Diffusion Image Prompt Embedding Reconstruction

Chinmaya Srivatsa, Varun Mulchandani and Dhruva Ungru Pulithaya

Project Info

Recently, AI leveraged image generation has taken the world by storm due to the advent of stable diffusion. Stable diffusion backed systems take prompts (text describing the image) as inputs which are then embedded into vectors and produce images from these embeddings.

This project is inspired by the Kaggle competition
https://www.kaggle.com/competitions/stable-diffusion-image-to-prompts/

In this project we attempt to reproduce/reconstruct vector embeddings of prompts that were used to generate images from stable diffusion.

Project Overview

We formulated preliminary approaches to reconstruct vector embeddings of stable diffusion prompts:

- First approach was to directly produce vector embeddings which would make it possible to build a model without a decoder
- Second approach was to build an encoder - decoder architecture model to re-generate prompts whose vector embeddings could then be calculated using a sentence transformer
- We compared our two approaches to a vision transformer baseline model that directly produced vector embeddings
- After these experiments, we utilized BLIP architecture to get the best results

Dataset Summary

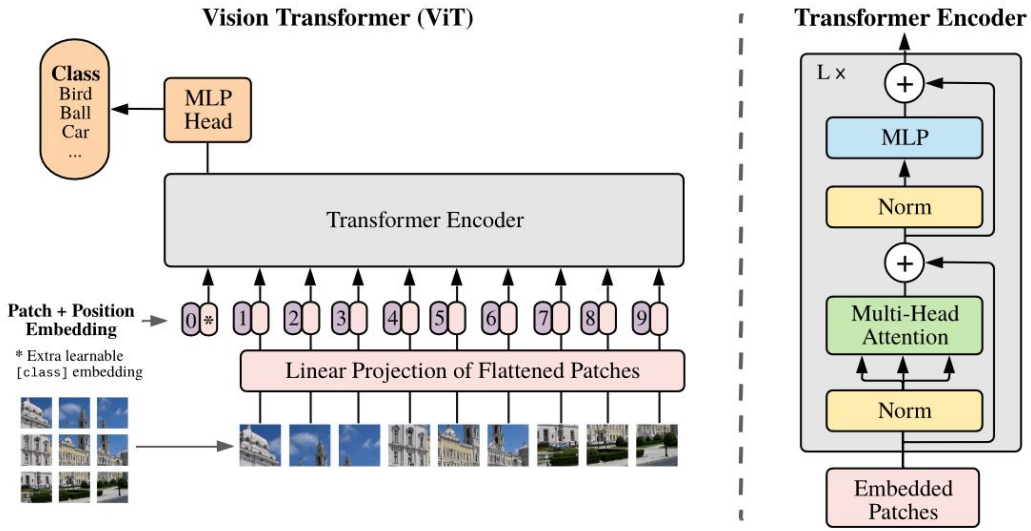
For this project we are using the DiffusionDB dataset.

- DiffusionDB is the first large-scale text-to-image prompt dataset
- It contains 14 million images generated by Stable Diffusion using prompts and hyperparameters specified by real users
- It is publicly available at https://huggingface.co/datasets/poloclub/diffusiondb
- DiffusionDB provides two subsets (DiffusionDB 2M and DiffusionDB Large) to support different needs.
- DiffusionDB Large is a superset of DiffusionDB 2M
- We are using a maximum of about 10% of the DiffusionDB in total for the purposes of this project

Baseline - Vision Transformer (ViT)

- Architecture:
  - Transformer encoder model
  - Pre-trained and fine-tuned on ImageNet-21k at 224x224 resolution
  - Images passed to model as 16x16 image patches which are linearly embedded

ViT - Architecture



Source : An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, Alexey Dosovitskiy*,†, Lucas Beyer*, Alexander Kolesnikov*, Dirk Weissenborn*, Xiaohua Zhai*

Model Training

- Training:
  - 10% of DiffusionDB (~150k samples) data used with 90 - 10 split for training and validation
  - Sentence Transformer used for embeddings of training and test prompts
  - AdamW optimizer used with an initial learning rate of 0.0001 with Cosine Annealing
  - Loss calculated using the Cosine Embedding Loss
  - Trained for 3 epochs with a batch size of 64

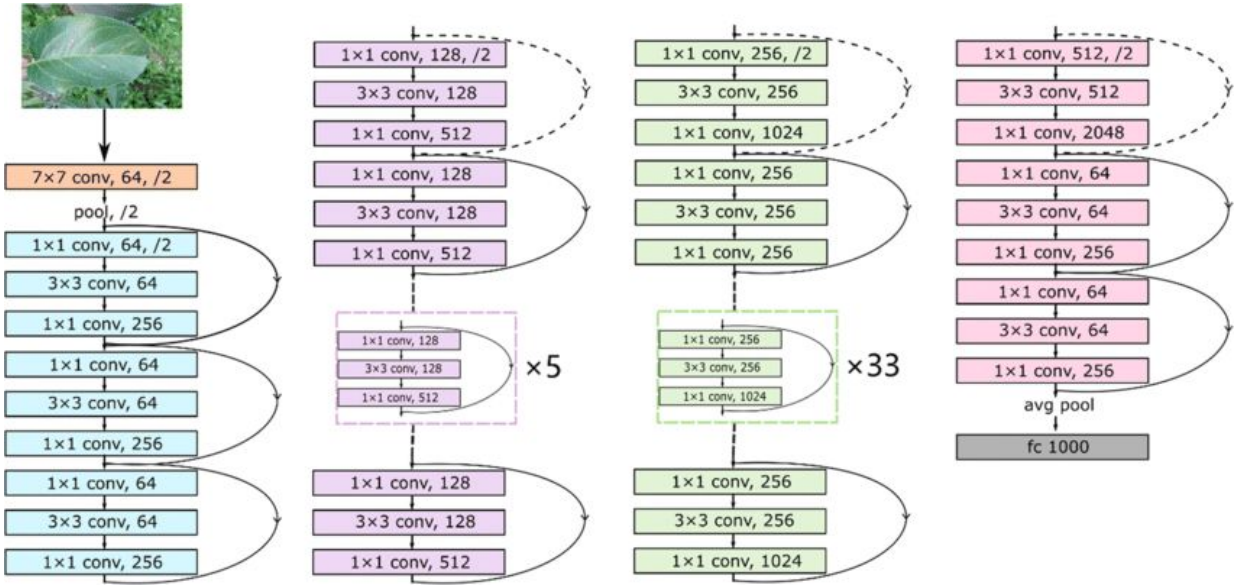
Model Evaluation

- Evaluation:
  - Validation Loss calculated by the Cosine Similarity
  - Performance measured by measuring Cosine Similarity with ground truth vector embeddings
  - Training loss measured after 3 epoch - 0.5013, Validation Cosine similarity measured - 0.4987
  - Test Cosine similarity measured - 0.5364

Approach 1 - CNN encoder using Transfer Learning

- Architecture:
    - ResNet152 pre-trained on ImageNet
    - Frozen Convolutional layers to be used as feature extractors
    - Classifier portion replaced with custom classifier
    - Custom classifier has 2 Fully Connected layers having 2048 and 384 neurons with a tanh activation layer and a dropout layer (with a dropout probability of 0.4) in between them

ResNet 152 - Architecture



Source : https://www.researchgate.net/publication/344004543/figure/fig3/AS:1137710379077632@1648262700055/Neural-network-architecture-of-ResNet152-The-dotted-shortcuts-increase-dimensions-and.png

Model Training

- Training:
  - 10% of DiffusionDB (~150k samples) data used with 85 - 15 split for training and validation
  - Sentence Transformer used for embeddings of training and testing prompts
  - AdamW optimizer used with an initial learning rate of 0.001 with Cosine Annealing
  - Loss calculated using the Cosine Embedding Loss
  - Trained for 3 epochs with a batch size of 64

Model Evaluation

Evaluation:

- Validation Loss calculated by the Cosine Embedding Loss
- Performance measured by measuring Cosine Similarity with ground truth vector embeddings
- Training loss measured after 3 epochs - 0.4721, Validation Cosine similarity measured - 0.5279
- Test Cosine similarity measured - 0.5317

Approach 2 - CNN Encoder, RNN Decoder

Architecture

- ResNet50 encoder, pre-trained on ImageNet
- LSTM decoder with attention

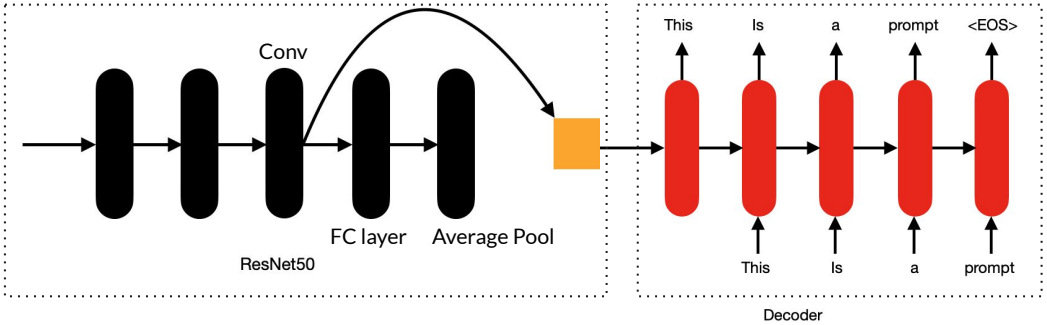
Dataset:

- 5% of the DiffusionDB dataset.

Model Architecture and Training

- Design:
    - We utilize the representations produced by ResNet50, specifically, the representations produced by it's third to last layer (i.e. the representation created by the convolutional layer right before the final Average Pool and FC layer).
    - We pass this representation into a LSTM based decoder, which produces the prompt tokens at each time step.
- Training:
    - Epochs - 5.
    - Optimizer - Adam.
    - Static Learning Rate - 3e-4.

Model Design

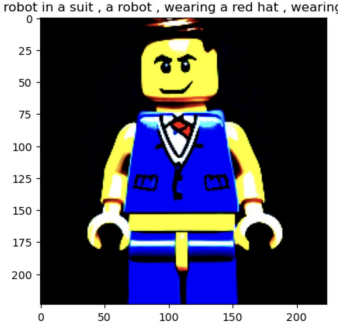


Results, Issues and areas of improvement

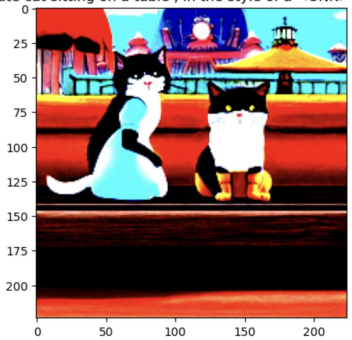
- Cosine similarity of 0.27 on test data.
- It can identify images with common objects like people, cats and dogs but it performs badly if the images did not contain these objects. Colors are identified sometimes.
- Potential areas for improvement:
  - Increased model capacity
  - Increase training data
- Ideally, we should utilize more data and more powerful pre-trained models.

Sample Results

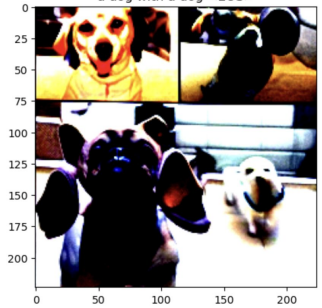
a robot robot in a suit , a robot , wearing a red hat , wearing a red hat ,



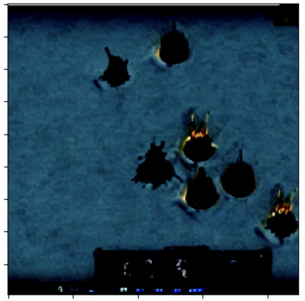
a cute cat sitting on a table , in the style of a <UNK> <EOS>



a dog with a dog <EOS>



a <UNK> of a <UNK> , <UNK> , <UNK> , <UNK> , <UNK> , <UNK> , <UNK> , <UNK> ,



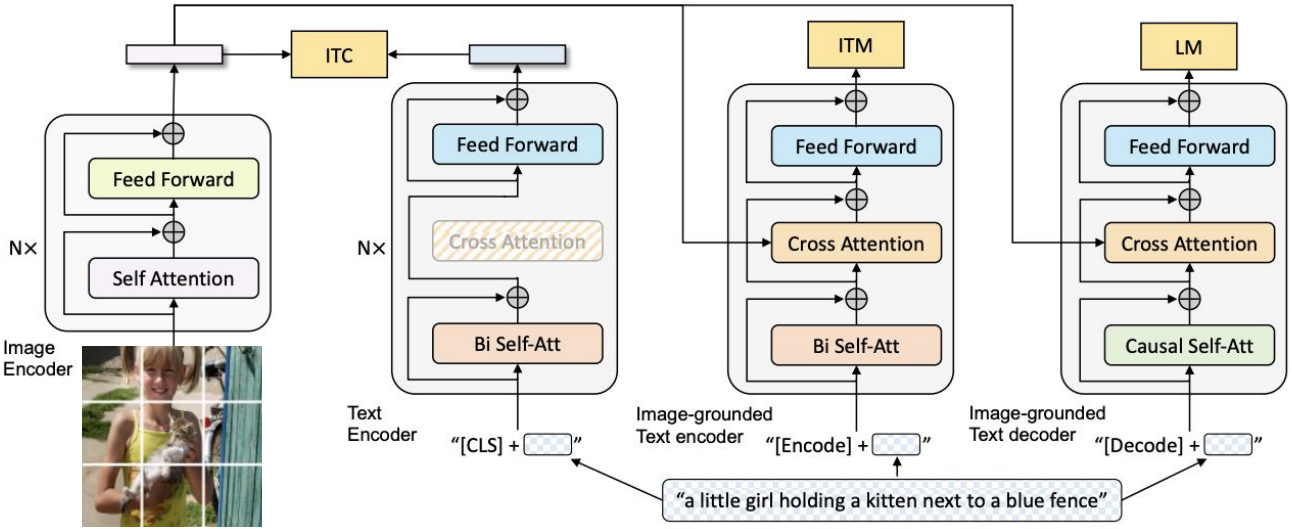
Challenges Encountered

- Huge dataset size; the DiffusionDB 2M dataset is nearly 2TB
- Porting dataset to locations outside Kaggle.
- High compute requirements
- Training the model takes a lot of time, even when utilizing NVIDIA P100 GPU instances available within Kaggle.

The best approach: BLIP

- A Unified Vision-Language Pre-processing technique
- Consists of Unimodal encoder, Image-grounded text encoder and Image grounded text decoder
- For our use case, we used only the image encoder and the Image grounded text decoder part of the model
- Computationally less expensive than a Encoder/Decoder or CNN based approach and requires lesser training than a simple ViT approach as it is pretrained.

Model Architecture



Source: https://github.com/salesforce/BLI

Model Training

Training:

- Pre-trained BLIP model available
- We use about 5% of the DiffusionDB dataset to train the model
- Learning rate of 1e-4, Batch size 4, 1 epoch
- Uses Cosine Annealing Learning rate scheduler

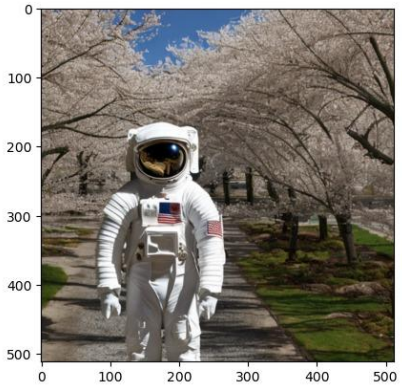
Model Evaluation

Testing:

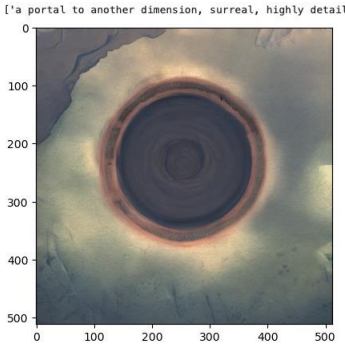
- Model tested on small subset of the DiffsuionDB dataset
- Cosine similarity: 0.639
- Performance measured by calculating the cosine similarity
- Model generates the text prompts directly. The Kaggle competition requires sentence embeddings, hence it is generated by passing the prompts to a sentence transformer

Sample Results

['astronaut doing the moon walk backwards']



['a portal to another dimension, surreal, highly detailed, artstation, smooth, sharp focus,']



['a robot drawing of a robot']