UNSW SYDNEY

Comp9417

Group work report

Stable Diffusion - Image to Prompts

ML and data mining

z5443003 | z5315804 | z5465340 | z5286124

Catalogue

SUMMARY 43

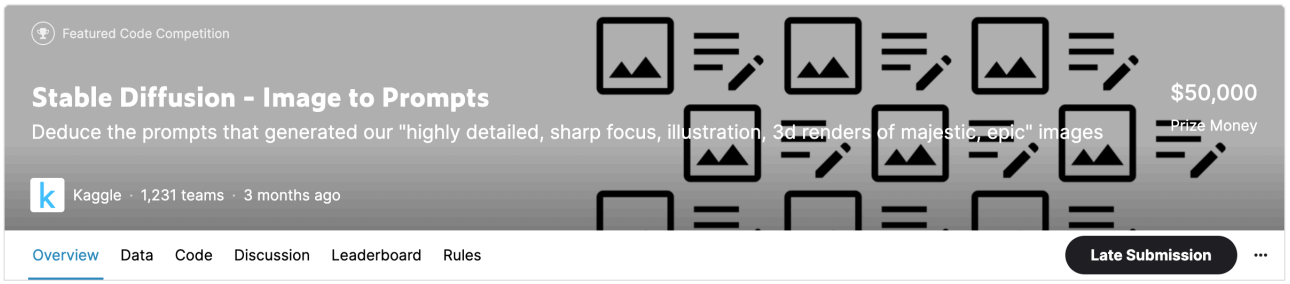
Part I

Topic

Stable Diffusion – Image to Prompts[1]

• Topic Introduction

Deduce the prompts that generated our "highly detailed, sharp focus, illustration, 3d renders of majestic, epic" images. This is a extremely tricky and challengeable topic.



What we will do? Destination?

The goal of this Topic is to reverse the typical direction of a generative text–to–image model: instead of generating an image from a text prompt, we will create a model which can predict the text prompt given a generated image, we will make predictions on a dataset containing a wide variety of `(prompt, image)` pairs generated by Stable Diffusion 2.0, in order to understand how reversible the latent relationship is.

What is our data? Data-resource?

Our task for this challenge is to predict the prompts that were used to generate target images. Prompts for this challenge were generated using a variety of (non disclosed) methods, and range from fairly simple to fairly complex with multiple objects and modifiers. Images were generated from the prompts using Stable Diffusion 2.0 (768–v–ema.ckpt) and were generated with 50 steps at 768x768 px and then downsized to 512x512 for the competition dataset. (This script was used, with the majority of default parameters unchanged.)

- **images/** – images generated from prompts; your task it to predict the prompt that was used to generate each image in this folder. The hidden re–run test folder contains approximately 16,000 images.
- **prompts.csv** – the prompts that were used to create the samples in the `images/` folder. These are provided as illustrative examples only. It is up to each competitor to develop their own strategy of creating a training

[1]https://www.kaggle.com/competitions/stable–diffusion–image–to–prompts/data

set of images, using pre-trained models, etc. Note that this file is not contained in the re-run test set, and thus referencing it in a Notebook submission will result in a failure.

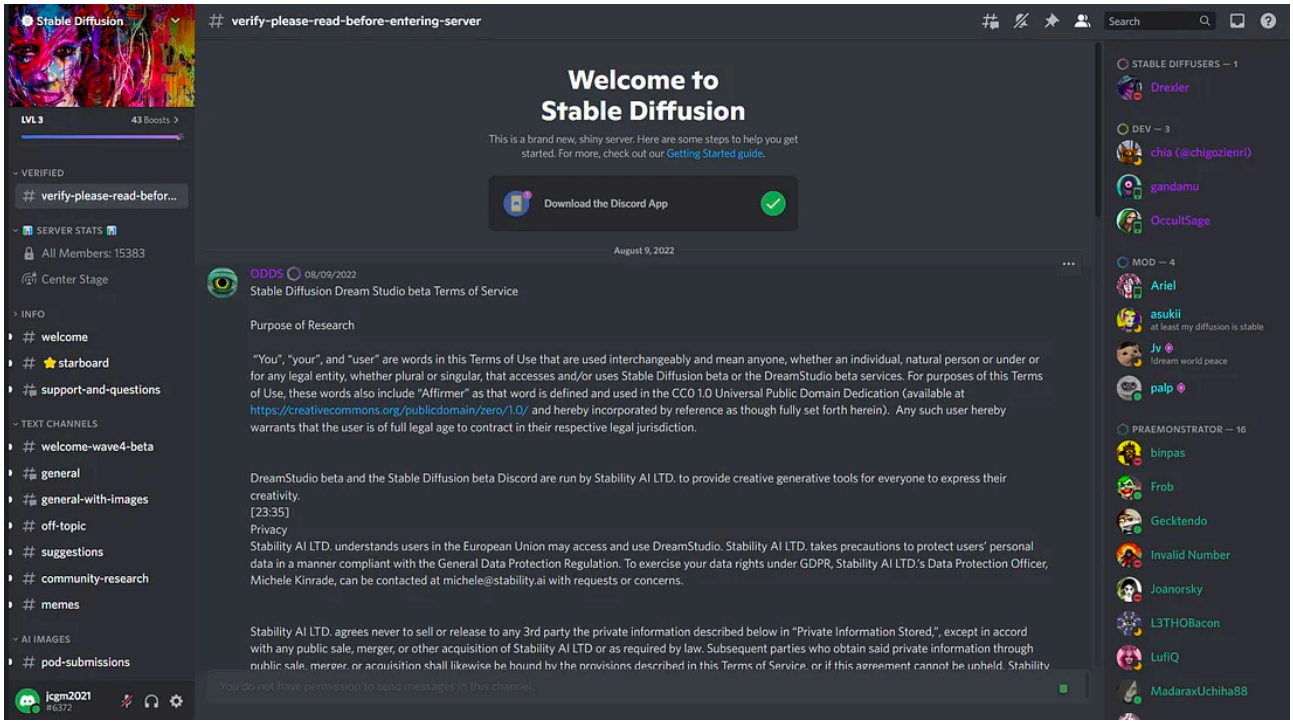
- **sample_submission.csv** – a sample submission file in the correct format. The values found in this file are embeddings of the prompts in the `prompts.csv` file and thus can be used validate your embedding pipeline. This notebook demonstrates how to calculate embeddings.

Why we choose this topic? Motivation?

The increasing popularity of text-to-image modeling has led to the emergence of a new field focused on cue engineering. With elements of both art and unresolved scientific inquiry, researchers and practitioners in machine learning are actively exploring the relationship between cues and the images they generate. Questions arise, such as whether adding "4k" to a cue is the most effective way to enhance its photographic quality, whether slight variations in cues lead to significant differences in the generated images, and how the order of cue keywords influences the resulting scenes. Our goal is to develop a model that can reliably reverse the diffusion process responsible for generating a given image.

To ensure a robust measure of cue similarity, even when character-level differences are present (e.g., comparing "epic cat" and "majestic kitten"), we will require participants to submit embedding vectors representing their predicted cues. Participants have the flexibility to directly model the embedding vectors or first generate the cues and then convert them into embeddings. We invite you to embark on this highly intriguing journey as we delve into the fascinating realm of text-to-image modeling.

Wait! But what is stable diffusion?



This is stable diffusion? No rush, let we watch more details as below.

What Is Stable Diffusion[2] ?

On Stability AI's website, Stable Diffusion is described as a text-to-image model that will enable billions of people to produce beautiful art in a matter of seconds.

Stable Diffusion sample images

his model employs a frozen CLIP ViT-L/14 text encoder to condition the model on text prompts, much like Google's Imagen does. The model uses a GPU with at least 10GB VRAM and is relatively lightweight with an 860M UNet and 123M text encoder.

[2] https://medium.com/codex/stable-diffusion-new-and-free-text-to-image-ai-tool-70f95ea14440

How Does It Work[3] ?

Stable Diffusion separates the image generating process into a “diffusion” process at runtime. Starting with only noise, it gradually improves an image until there is no noise left at all, bringing it more and closer to a provided text description.

Over the course of a month, Stability AI trained Stable Diffusion on a cluster of 4,000 Nvidia A100 GPUs operating in Amazon Web Services. Ludwig Maximilian University of Munich’s CompVis machine vision and learning research group directed the training, and Stability AI provided the computing resources.

Images by Jim Clyde Monge

Through its Discord server, Stability AI has made the Stable Diffusion model accessible to a select group of users.

[3] https://medium.com/codex/stable-diffusion-new-and-free-text-to-image-ai-tool-70f95ea14440

Part II

distribution

• Teamwork distribution and plan

Tasks	Member	Details
Modeling	2 people (z5286124) (z5315804)	• CLIP + GPT−2 • ResNet_LSTM + GPT−2
Analying	1 person (z5465340)	• Evaluation of the machine learning model itself • Assessment of training effectiveness
Reporting	1 person (z5443003)	• All parts of report and layout

We will obey this task distribution to complete our tasks.

	Modeling	Analying	Report
First week 7.20 due	1. The overall framework, training logic, training process 2. Code framework (function name) is basically complete	1. Horizontal analysis of the model itself 2. Framework for vertical analysis of the model 3. Introduction and rationale for the methodology of the analysis	1. Overall framework (3 levels of headings) • Background • Reasons for choosing the topic • Significance of Selection • Introduction of each model 2.Final draft framework
Second Week 7.24 due	Various tests can run results and provide next steps to improve model accuracy	Give basic analyses and improvements based on the initial framework	70% populated with content based on the initial framework
Second Week 7.27 due	Efforts to improve model accuracy (all done)	Analysis and discussion of results	Layout optimisation, design of illustrations + design of cover, optimisation of overall report logic, completion of 90% of report content
Third Week 8.2 due	Optimise model + tweak typography + annotate code	Modification of the overall details of the assessment component	Typography optimisation + detail modifications and touch−ups

We will obey this plan to complete our task. At due time we will gather together having a meeting to exchange progress and solutions for challenges.

• Model pre-selection

Our group will choose 3–4 of following models to train our dataset.

Highly difficult model

Convolutional Neural Networks (CNN) : CNN is one of the most commonly used models for image classification. It extracts features of an image through multiple convolutional and pooling layers and uses a fully connected layer for classification.CNNs have high accuracy but require large amounts of data and computational resources for training.

Deep Residual Networks (ResNet): ResNet is a deep neural network architecture that solves the problem of gradient vanishing in deep network training through residual connectivity.ResNet performs well on image classification tasks, especially when dealing with large-scale datasets and deeper network structures.

Attention Mechanism: Attention Mechanism models achieve selective attention to image features by learning the attention weights of different regions in an image. These models can more accurately capture important information in images and achieve better performance in image classification tasks.

less difficult models

Naive Bayes: Naive Bayes is a classification algorithm based on Bayes' theorem, which assumes that features are independent of each other and uses prior and conditional probabilities for classification.

Logistic Regression: Logistic regression is a generalised linear model commonly used in binary classification problems. It performs classification by multiplying input features with weights and using a logistic function to map the result to a probability value between [0,1].

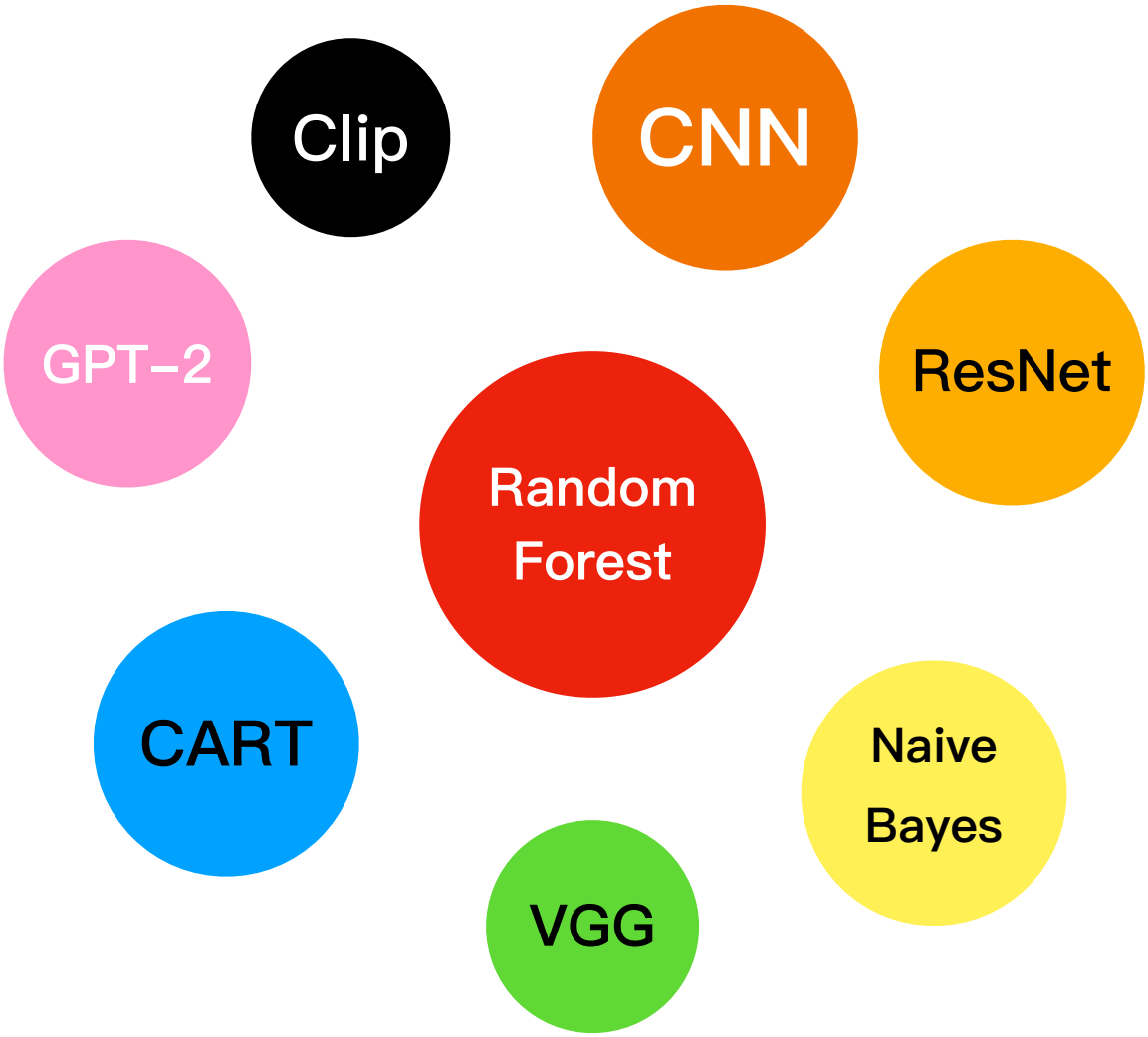
Support Vector Machines (SVM) : SVM is a classical supervised learning algorithm that can be used for image classification tasks. It performs classification by mapping image features to a high-dimensional space and constructing an optimal hyperplane.SVM works better with small datasets.

Decision Trees: Decision Trees are a classification model based on a tree structure that can be used for image classification. It classifies images through a series of split conditions and decision nodes. Decision Trees are easy to understand and implement, but may be less accurate when dealing with complex datasets.

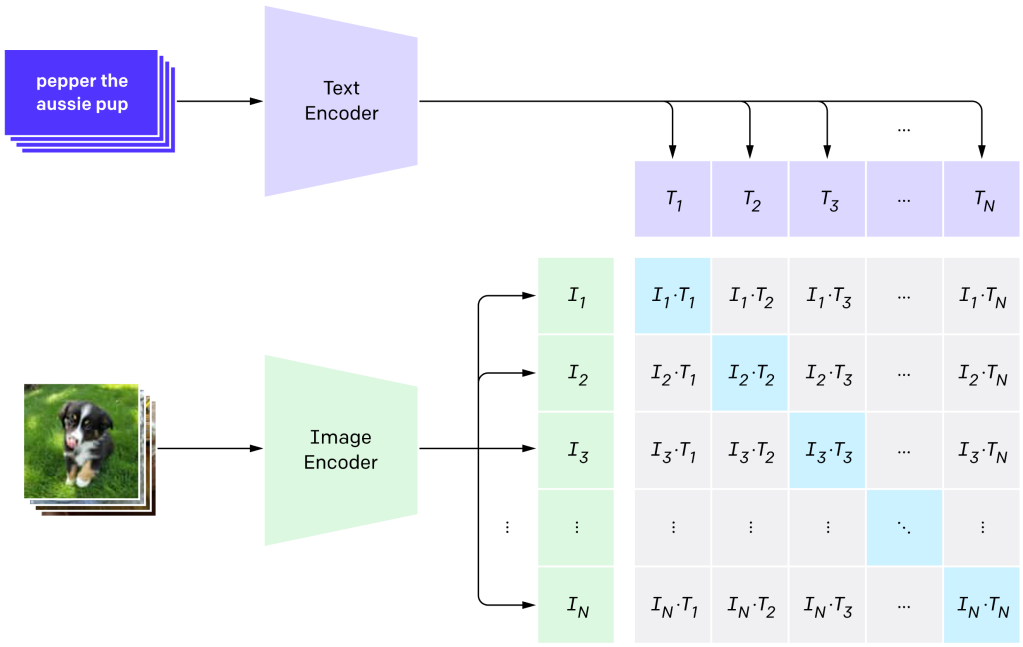
Random Forests (Random Forests) : Random Forests is an integrated learning algorithm that integrates multiple decision trees for classification. It performs well in multi-category problems and when dealing with large-scale datasets.

CLIP Model – ViT-B/32

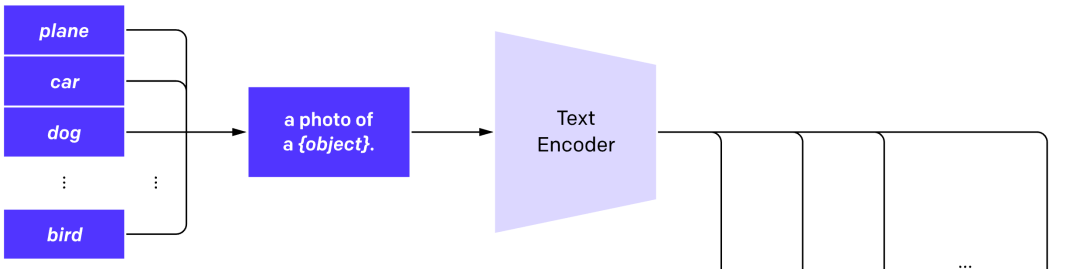
After discussion, what models we will use?



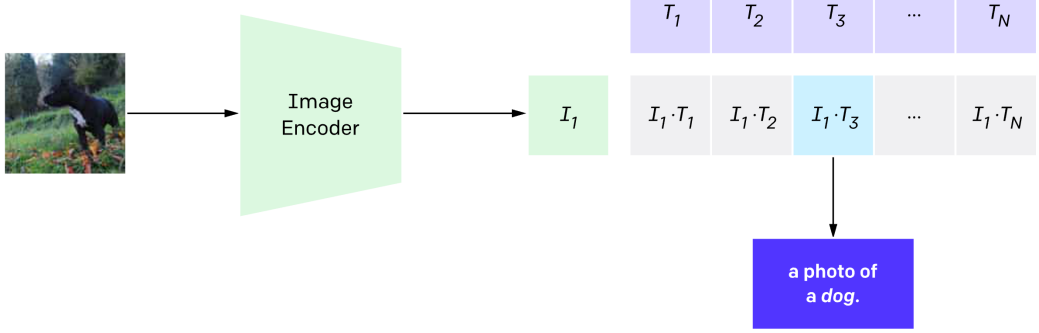
1. Contrastive pre-training



2. Create dataset classifier from label text



3. Use for zero-shot prediction



• Model Final selection

Discussion

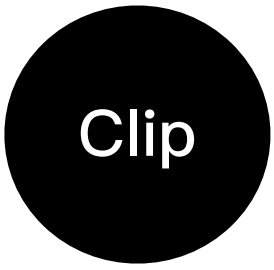
Comprehensively consider our task, a suitable method should be specially designed for sequence–to–sequence tasks, such as language models, transformers, or encoder–decoder architectures. These models can easily and effectively capture the characteristics of language and sequences so that to generate a meaningful prompt. In addition, pre–trained models, such as GPT–2 and CLIP, show promising results in image description and text–to–image retrieval tasks and can be used as a starting point for this particular task.

If we use Attention Mechanism?

- Attention Mechanism is widely used for tasks such as machine translation and image description. It allows the model to focus on relevant parts of the input when generating output.
- Attention Mechanism can be used for text–to–image retrieval, but it is only one component of a more comprehensive model. It is not sufficient by itself to perform the entire task of generating textual cues from images.

If we use Naive Bayes? Support Vector Machines (SVM)? Logistic Regression?

- Naive Bayes, SVM and Logistic Regression are traditional machine learning algorithms used for classification tasks. They are not applicable to text–to–image retrieval because the task involves generating natural language text, which is sequential in nature and has complex relationships between words.
- The inability of these algorithms to capture sequential dependencies or linguistic semantics makes them less suitable in text generation tasks.



Result

Lastly, we choose CLIP and ResNet_Lstm model to train our dataset, CLIP is a multimodal model that can process both image and text inputs. ResNet is a classical model of convolutional neural networks that performs well on image classification tasks.

Why do we choose this two models？ Attention?

	Advantages ヾ(@^▽^@)ノ	Attention (`●__●´)/
CLIP	① ★**Multimodal learning**★ CLIP is a multimodal model that can process both image and text inputs. It achieves a tight correlation between images and text by learning the common representation space of images and text, which makes it excellent at predicting relationships between images and text. ② ★**Generalisation across datasets**★ CLIP is pre-trained on large-scale datasets, which makes it powerful in generalisation. It can perform well on different datasets without the need to retrain for each specific dataset. ③ ★**Efficient Feature Representation**★ The image and text representations learnt by CLIP are very meaningful at the semantic level, making it able to better understand the relationship between images and text for effective prediction.	① ★**Model complexity**★ both CLIP and ResNet_Lstm are large-scale models that may require larger computational resources and longer training times. With limited hardware and time budgets, there may be a trade-off between model size and performance. ② ★**Dataset limitations**★ while CLIP is pre-trained on large-scale datasets, the model may have challenges in generalisation performance if the data provided by the
ResNet_LSTM	① ★**Powerful image feature extraction**★ ResNet_Lstm is the classic model of convolutional neural networks, and it excels at image classification tasks. By using pre-trained ResNet_Lstm models, you can obtain advanced feature representations of generated images that provide more information about the image and help match it with text. ② ★**Open source and pre-trained models**★ The open source implementation of ResNet_Lstm and the availability of pre-trained models make it easy to integrate ResNet_Lstm into your models and speed up the experimentation and development process.	

Part III

Implementation

• Feature Engineer

the feature engineering in both models involves using pre-trained models for image encoding and text tokenization, as well as proper data preprocessing and batching techniques to prepare the data for training the joint image-caption model. The use of CLIP for image encoding and GPT-2 for text generation allows the model to learn meaningful joint representations of images and text, enabling it to generate captions for images.

Model 1 - CLIP+GPT-2

- CLIP (Contrastive Language-Image Pretraining) Model: The function uses the CLIP library to load pre-trained models for image encoding. CLIP is a neural network model that learns joint representations of images and their associated text descriptions.
- Image Preprocessing: The function loads the image data from the provided dataset and preprocesses it using the CLIP model's preprocessing steps. This includes resizing and normalizing the image data.
- Image Encoding: The function uses the selected CLIP model to encode each image in the dataset. The image embeddings are generated by passing the preprocessed images through the CLIP model, which maps images into a vector space.
- Prompt Splitting: The function extracts and stores the text prompts corresponding to each image in the dataset.
- Data Saving: The function saves the generated image embeddings and corresponding prompts to a file for later use during training.
- GPT-2 Tokenization: The class uses the GPT-2 tokenizer to preprocess the text prompts. It tokenizes the text and converts it into input_ids, which are used for training the GPT-2 language model.
- Padding and Truncation: To handle variable-length sequences, the text input_ids are padded or truncated to a fixed length (maximum sequence length) so that all samples have the same size. This ensures that the model can process the data efficiently in batches.
- Masking: The class also creates an attention mask for the input_ids to indicate which tokens are actual text and which ones are padding. This helps the model ignore padding tokens during training.

- Normalization: Optionally, the image embeddings can be normalized to unit norm. This process ensures that all embeddings have the same length, making them more suitable for certain similarity-based tasks.
- Data Caching: To avoid recomputing the embeddings and tokenization on each epoch during training, the class caches the image embeddings, tokenized text, and maximum sequence length in a pickle file.

Model 2 - ResNet_LSTM + GPT-2

Data Loading and Preprocessing:

- The ImageDataset class is defined to handle loading and preprocessing of image and prompt data.
- Two sets of data loaders are created using DataLoader for the training and validation datasets.
- Images are transformed using standard image augmentation techniques (e.g., random rotation, random crop, horizontal and vertical flips, color jittering, normalization).
- The prompt texts are tokenized using the GPT-2 tokenizer, adding special tokens, padding, and generating attention masks for variable-length sequences.

Model Architecture:

- The Resnet_LSTM class defines the model architecture, combining the ResNet-50 model and the GPT-2 model.
- The ResNet-50 model is used to extract image features, and a fully connected layer (FC) is applied to reduce the feature dimension to the specified embedding dimension.
- The LSTM layer is used to process the image features before inputting them into the GPT-2 model.
- The GPT-2 model is used to generate text captions based on the input image features.

Training and Validation:

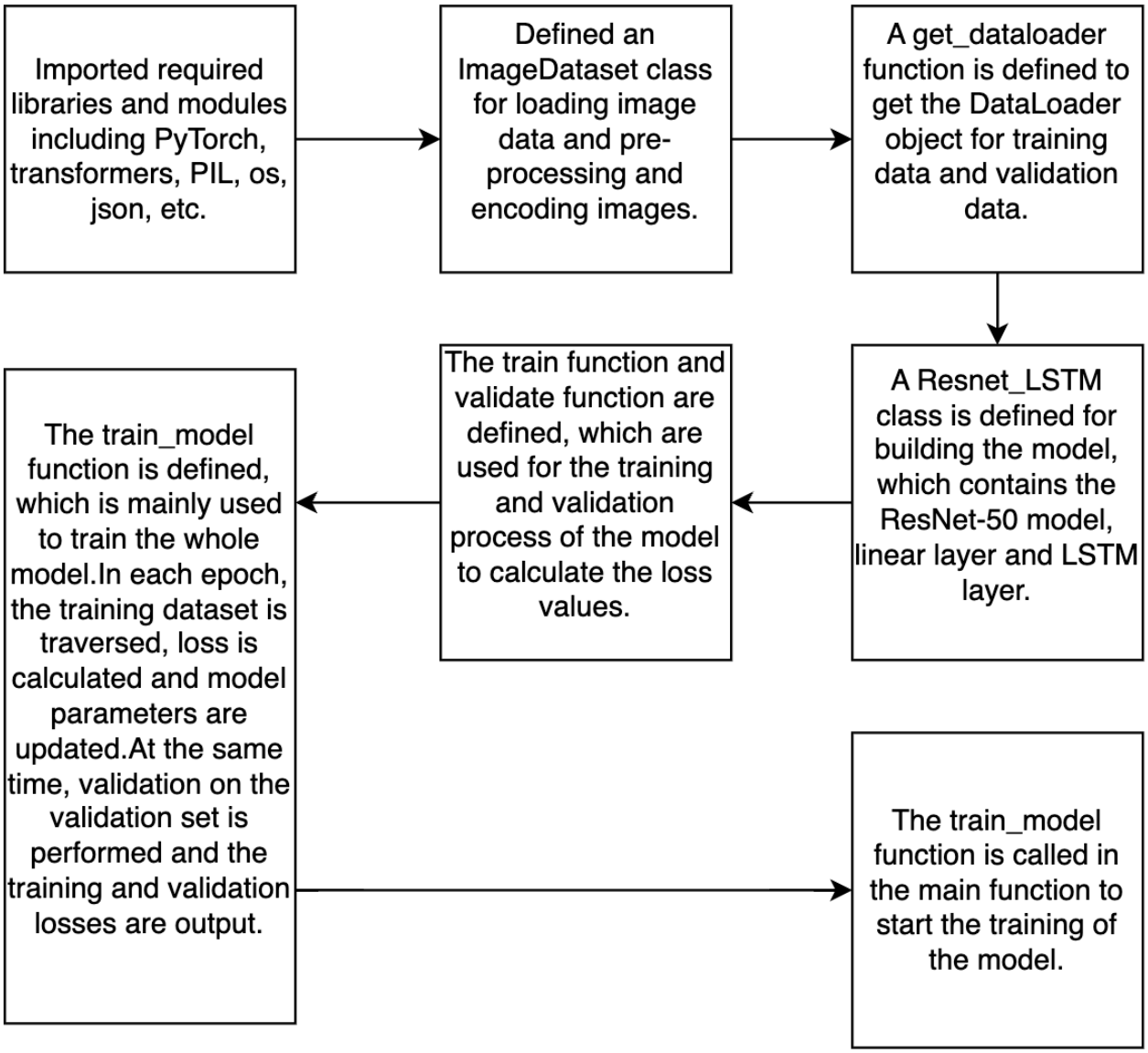
- The train function is defined to perform training for one epoch. It calculates the loss and backpropagates the gradients.
- The validate function is used for validation, calculating the loss on the validation dataset without updating the model parameters.

- The train_model function orchestrates the training process, iterating over the specified number of epochs and updating the model parameters based on training loss.

• Model Design and Implementation

In this part, I will use simplified words to describe our logic and code structure with parameter. Let we figure out it step by step.

The Process of ResNet_LSTM + GPT-2



① It combines the pre-trained GPT-2 model and the ResNet-50 model. The main process is as follows:

② Import the required libraries and modules, including PyTorch, transformers, PIL, os, json, etc.Defined an ImageDataset class for loading image data and pre-processing and

encoding images.A get_dataloader function is defined to get the DataLoader object for training data and validation data.A Resnet_LSTM class is defined for building the model, which contains the ResNet-50 model, linear layer and LSTM layer.

③ The train function and validate function are defined, which are used for the training and validation process of the model to calculate the loss values.

④ The train_model function is defined, which is mainly used to train the whole model. In each epoch, the training dataset is traversed, loss is calculated and model parameters are updated. At the same time, validation on the validation set is performed and the training and validation losses are output.

⑤ The train_model function is called in the main function to start the training of the model.

The Logic of ResNet_LSTM + GPT-2

The overall logic is that the input image is feature extracted by the ResNet-50 model and the features are mapped to the input dimensions of the GPT-2 model by a linear layer. Then, the features are processed through an LSTM layer to generate the initial text of the image description. Then, the initial text is further processed using the GPT-2 model to generate the final image description. The Adam optimiser and learning rate scheduler are used during the training process in order to facilitate the convergence of the model and optimisation of the training results.

Model Training of ResNet_LSTM + GPT-2 with parameters

- **ImageDataset class:**
  - **Parameters**:
    - data_dirs: List of directories containing image dataset.
    - mode: Image preprocessing mode, default is 0.
  - **Initialization**:
    - Load the GPT-2 tokenizer.
    - Load image files and corresponding text descriptions.
    - Define the image preprocessing transforms.
  - **getitem method:**
    - Get the image and its corresponding text description.
    - Perform GPT-2 tokenization on the text.

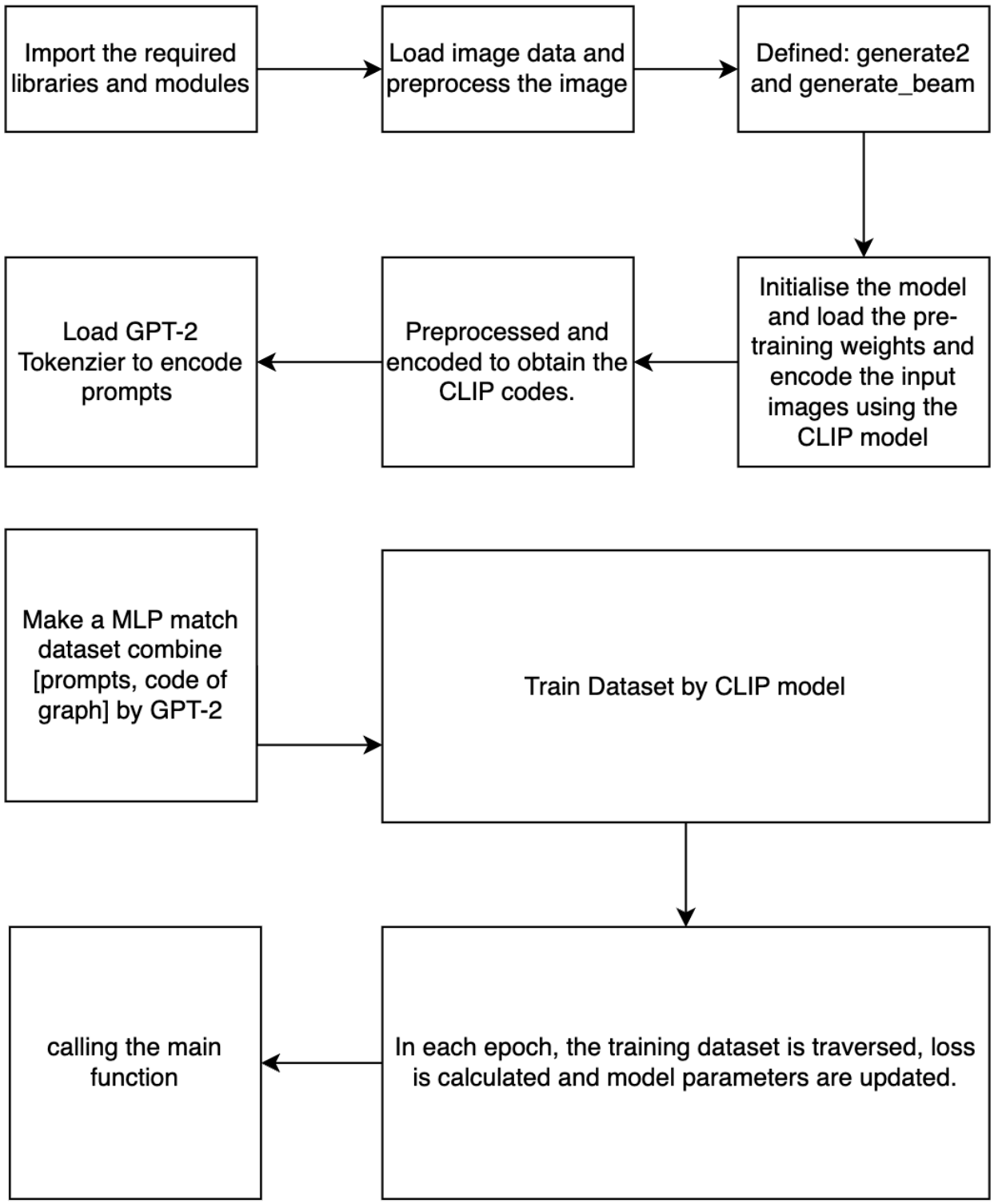
    - Return image tensor, input_ids of text, and attention mask.
- **get_dataloader function:**
    - **Parameters**:
        - None
    - **Function**:
        - Define the path to the image dataset.
        - Create data loaders for the training set and validation set.
    - **Return**:
        - Data loaders for the training set and validation set.
- **Resnet_LSTM class:**
    - **Parameters**:
        - resnet_model: Pre-trained ResNet-50 model.
        - gpt2_model: Pre-trained GPT-2 model.
        - embedding_dim: LSTM output embedding dimension.
    - **Initialization**:
        - Define the ResNet-50 model, a linear layer, and an LSTM layer.
        - Load the GPT-2 model.
- **train function:**
    - **Parameters**:
        - **model:** Resnet_LSTM model for training.
        - **dataloader:** DataLoader for the training dataset.
        - **optimizer:** Optimizer for updating model parameters.
        - **device:** Device (GPU or CPU).
        - gpt2_model: Pre-trained GPT-2 model.
        - scheduler: Learning rate scheduler for adjusting learning rate.
    - **Function**:
        - Set the model to training mode.
        - Iterate over the training dataset to get images and text.
        - Pass images and text to the Resnet_LSTM model, compute loss.
        - Backpropagate, update model parameters and learning rate.
        - Return the average training loss.
- **validate function:**

- **Parameters**:
  - **model:** Trained Resnet_LSTM model.
  - **dataloader:** DataLoader for the validation dataset.
  - **device:** Device (GPU or CPU).
  - **gpt2_model:** Pre-trained GPT-2 model.
- **Function**:
  - Set the model to evaluation mode.
  - Iterate over the validation dataset to get images and text.
  - Pass images and text to the Resnet_LSTM model, compute loss.
  - Return the average validation loss.

- **train_model function:**
  - **Parameters**:
    - None
  - **Function**:
    - Define training parameters and hyperparameters.
    - Load the GPT-2 tokenizer and pre-trained GPT-2 model.
    - Load the pre-trained ResNet-50 model.
    - Create data loaders for the training set and validation set.
    - Create the Resnet_LSTM model and move it to the device.
    - Define the optimizer and learning rate scheduler.
    - Loop over epochs for training, output the training and validation losses, and save the model.
- **Main Program:**
  - Call the **train_model** function to start training.

The Process of CLIP + GPT-2

It uses a combination of the CLIP model and the GPT-2 model to generate a description of the image. The main process is as follows:



① Import the required libraries and modules including PyTorch, CLIP model, GPT-2 model, data processing, etc.Defined an ImageDataset class for loading image data and pre-processing the image.Two functions for generating image descriptions are defined: generate2 and generate_beam. these functions use the GPT-2 model to generate textual descriptions related to the image based on the input image and prompt text.The main function is defined to initialise the model and load the pre-training weights and encode the input images using the CLIP model.

② The image dataset is loaded and the images are preprocessed and encoded to obtain the CLIP codes and text data.

③ A makeDataset class is defined to build the training dataset. The class processes the image and text data with appropriate padding operations for model training.

④ An evaluate function is defined for evaluating the model on the validation set and calculating the loss.

⑤ The train function was defined for the training process of the model. In each epoch, the training dataset is traversed, loss is calculated and model parameters are updated.

⑥ By calling the main function, the model and data are loaded and the training process is performed.

The Logic of CLIP + GPT-2

The overall logic is that the input images are encoded into feature vectors by the CLIP model, which is then combined with the GPT-2 model to generate the corresponding image descriptions based on the input cue text. The training dataset contains images and corresponding text descriptions, and the model learns these samples to gradually improve the accuracy and semantic representation of the image descriptions. An optimiser and a learning rate scheduler are used during the training process, while the loss is saved during the training process for subsequent analysis and tuning.

Model Training of CLIP + GPT-2 with parameters

- **ImageDataset class:**
  - **Parameters:**
    - data_dirs: List of directories containing the image dataset.
    - mode: Image preprocessing mode, default is 0.
  - **Initialization:**
    - Load the GPT-2 tokenizer.
    - Load image files and corresponding text descriptions.
    - Define the image preprocessing transforms.
  - **getitem method:**
    - Get the image and its corresponding text description.
    - Perform GPT-2 tokenization on the text.
    - Return image tensor, input_ids of text, and attention mask.
  - **get_dataloader function:**
    - **Parameters:**
      - None
    - **Function:**
      - Define the path to the image dataset.
      - Create data loaders for the training set and validation set.
    - **Return:**
      - Data loaders for the training set and validation set.
- **Resnet_LSTM class:**
  - **Parameters:**
    - resnet_model: Pre-trained ResNet-50 model.
    - gpt2_model: Pre-trained GPT-2 model.
    - embedding_dim: LSTM output embedding dimension.
  - **Initialization:**
    - Define the ResNet-50 model, a linear layer, and an LSTM layer.
    - Load the GPT-2 model.
  - **train function:**
    - **Parameters:**
      - model: Resnet_LSTM model for training.
      - dataloader: DataLoader for the training dataset.

    - optimizer: Optimizer for updating model parameters.
    - device: Device (GPU or CPU).
    - gpt2_model: Pre-trained GPT-2 model.
    - scheduler: Learning rate scheduler for adjusting the learning rate.
  - **Function:**
    - Set the model to training mode.
    - Iterate over the training dataset to get images and text.
    - Pass images and text to the Resnet_LSTM model, compute loss.
    - Backpropagate, update model parameters, and learning rate.
    - Return the average training loss.
- **validate function:**
  - **Parameters:**
    - model: Trained Resnet_LSTM model.
    - dataloader: DataLoader for the validation dataset.
    - device: Device (GPU or CPU).
    - gpt2_model: Pre-trained GPT-2 model.
  - **Function:**
    - Set the model to evaluation mode.
    - Iterate over the validation dataset to get images and text.
    - Pass images and text to the Resnet_LSTM model, compute loss.
    - Return the average validation loss.
- **train_model function:**
  - **Parameters:**
    - None
  - **Function:**
    - Define training parameters and hyperparameters.
    - Load the GPT-2 tokenizer and pre-trained GPT-2 model.
    - Load the pre-trained ResNet-50 model.
    - Create data loaders for the training set and validation set.
    - Create the Resnet_LSTM model and move it to the device.
    - Define the optimizer and learning rate scheduler.
    - Loop over epochs for training, output the training and validation losses, and save the model.

• Model Optimisation

Optimisation of CLIP + GPT-2

This code defines a deep learning model that combines a ResNet-50 image classifier and an LSTM language model with a GPT-2 transformer. It also includes a custom dataset and training loop for generating image captions using the combined model.

The code does the following:

- Defines a custom Resnet_LSTM class that combines the ResNet-50 model and the GPT-2 model. It takes an image as input, extracts features using ResNet-50, passes them through a linear layer, feeds the result to the LSTM, and finally processes the LSTM output through GPT-2 to generate captions.
- Implements an ImageDataset class that loads image data from a directory, preprocesses them using transformations, and returns images along with their filenames.
- Defines a load_model function to load the pretrained ResNet-50 and GPT-2 models and initialize the combined Resnet_LSTM model with their weights.
- Implements a generate_and_save_predictions function that generates captions for images using the trained model and saves the results to a text file in GPT-2 tokenized format.
- In the main() function, it loads the pretrained model from a checkpoint file, sets up the dataset and dataloader for the test images, and generates and saves captions for each image.

Optimisation of ResNet_LSTM + GPT-2

This code defines a training pipeline for a ClipCaption model that combines CLIP and GPT-2 models for image captioning. The pipeline includes data preprocessing, model definition, training, and evaluation.

The code does the following:

- Implements an image_encoder function that takes an image dataset and encodes each image using the CLIP model, creating a dictionary of image embeddings indexed by image IDs.
- Defines a custom makeDataset class that loads image embeddings and GPT-2 tokenized captions from files, pads the captions, and returns them as tensors.

- Creates a ClipCaptionModel class that combines CLIP and GPT-2 models to generate captions for images. It takes CLIP image embeddings and GPT-2 tokenized captions as inputs and predicts the next word in the caption sequence.
- Defines a training function that performs training on the combined model using AdamW optimizer and linear learning rate scheduler.
- In the main() function, it sets up the training data, initializes the model, and starts the training process. It saves the model's training loss to a file after each epoch.

Hyperparameter optimization

• batch_size: An increase in batch size speeds up the training of the model, but a batch size that is too large can make it difficult for the model to converge. Too small a batch size can lead to inaccurate gradient changes. In our test of 4 hours length, the batch size is considered to be 40 for a dataset with 1k data size, and 50 for a dataset with 5k data size.

• Optimizer: Since the model needs to deal with sparse gradient problem, we can choose Adam or AdamW optimizer. Due to the limitation of the device, we want the model to converge quickly to save some time. So we choose AdamW optimizer to accelerate the convergence speed.

• Schedule: In order to get a good training model, we add scheduler as the model optimization. In the choice of scheduler, we choose linear preheating scheduler. the preheating parameters of this method have been adjusted for many rounds, and by observing the loss curves, we found that the loss mainly jittered at the beginning of training. So, we choose the first 10% of all STEPs as the value of the preheating parameter. Due to the equipment limitation and dataset size, our initial learning rate is not very high, we set it to 1e-4. we hope to dynamically adjust the learning rate by scheduler to achieve the best result.

• For the image processing part, we choose ViT and ResNet for comparison. We expect the larger model can extract more information and features of the pictures.

• For the language disambiguator selection, we try bert disambiguator and gpt2 disambiguator. We found that the language predicted by bert's lexicon needs further processing and the result is not good. So, we switched to GPT2 for prompt encoding.

• For the choice of language model, we believe that GPT2 is better than other models in predicting text from pictures. The reason is that GPT2 covers a wide range of domains, and

the content of our images is also diverse. So we want the model to predict as much content as possible.

- We have done normalization on the images, due to the presence of mas

Part IV

Evaluation

• Comparison of different Evaluation methods

We have many metrics to evaluate our model and training performance, like as follows:

ROC

The ROC curve is a graph with True Positive Rate (also known as Recall Rate or Sensitivity) as the vertical axis and False Positive Rate (FPR) as the horizontal axis. It plots the relationship between the model's Recall Rate and False Positive Rate at different thresholds.The ROC curve helps us to weigh the model's Recall Rate and False Positive Rate at different thresholds and choose the right threshold to achieve our desired performance.

AUC

AUC is the area under the ROC curve, i.e., the area enclosed by the ROC curve and the axes.The value of AUC ranges from 0.5 to 1, where 0.5 indicates that the model's performance is comparable to a random guess, while 1 indicates that the model performs perfectly and is able to distinguish between positive and negative examples perfectly.The closer the AUC is to 1, the better the model's performance is.

Confusion Matrix

Confusion Matrix: the Confusion Matrix is a two-dimensional matrix used to visualise the performance of a classification model. It compares the predictions of the model with the true labels and can calculate metrics such as accuracy, recall and precision. The four key elements of the confusion matrix are as follows:

- True Positive (TP): the number of positive cases predicted by the model that are actually positive.
- True Negative (TN): the number of cases that the model predicts as negative and that are actually negative.
- False Positive (FP): the number of positive cases predicted by the model that are actually negative.
- False Negative (FN): the number of negative cases predicted by the model that are actually positive.

Accuracy

Accuracy is one of the most commonly used evaluation metrics for classification models, which measures the proportion of all samples that the model correctly classifies. The accuracy rate is calculated as:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Where TP (True Positive) denotes the number of true examples, TN (True Negative) denotes the number of true negative examples, FP (False Positive) denotes the number of false positive examples, and FN (False Negative) denotes the number of false negative examples.

Recall

Recall rate is also known as sensitivity, true case rate, which measures the ability of the model to recognise positive case samples. Recall rate is calculated by the formula:

Recall = TP / (TP + FN)

Precision

Precision: Precision measures the proportion of true cases in the sample predicted to be positive by the model. The formula for calculating Precision is:

Precision = TP / (TP + FP)

F1–score

F1–score (F1–value): the F1–value is the reconciled average of Precision and Recall, which combines the accuracy and completeness of the classification model.The F1–value is calculated as:

F1–score = 2 * (Precision * Recall) / (Precision + Recall)

Ratio

Ratio is one of the most common text similarity metrics. It uses the Levenshtein edit distance (also known as edit distance or minimum edit distance) to calculate the similarity between two texts.

Edit distance is the minimum number of operations to convert one string to another by inserting, deleting and replacing characters.

The similarity score used by Ratio to calculate this is: Similarity Score = (Total Characters – Edit Distance) / Total Characters.

This score ranges from 0 to 1, with 1 indicating a perfect match and 0 indicating that the two texts are completely different.

Quick Ratio

Quick Ratio is a variant of the similarity metric, very similar to Ratio, but with some tweaks to improve performance when calculating edit distances.

Quick Ratio uses an algorithm called "Quick Edit Distance" to approximate the Levenshtein edit distance. Instead of calculating the edit distance completely, Quick Ratio calculates an approximate edit distance by considering only substitution operations.

Quick Ratio is calculated similarly to Ratio:

$$Similarity\ Score = \frac{Total\ Characters - Quick\ Edit\ Distance}{Total\ Characters}$$

Quick Edit Distance is usually computed faster than Full Edit Distance, so Quick Ratio may be more efficient when working with large amounts of text.

Cosine similarity

Cosine similarity is a metric used to calculate the similarity between two vectors and is commonly used for text similarity comparison.

In text similarity, text is considered as a collection of word vectors and can be represented as a combination of word vectors.

Cosine similarity measures the similarity between two text vectors by calculating the angle between them. The smaller the angle is, the closer the cosine similarity is to 1, which means the more similar the two texts are; the larger the angle is, the closer the cosine similarity is to 0, which means the less similar the two texts are.

$$\text{Cosine\_Similarity}(A,\ B) = \frac{A \cdot B}{|A| \cdot |B|}$$

Here we will use three models to evaluate our train performance, they are namely Ratio, Quick Ratio, Cosine Similarity.

• Output sample

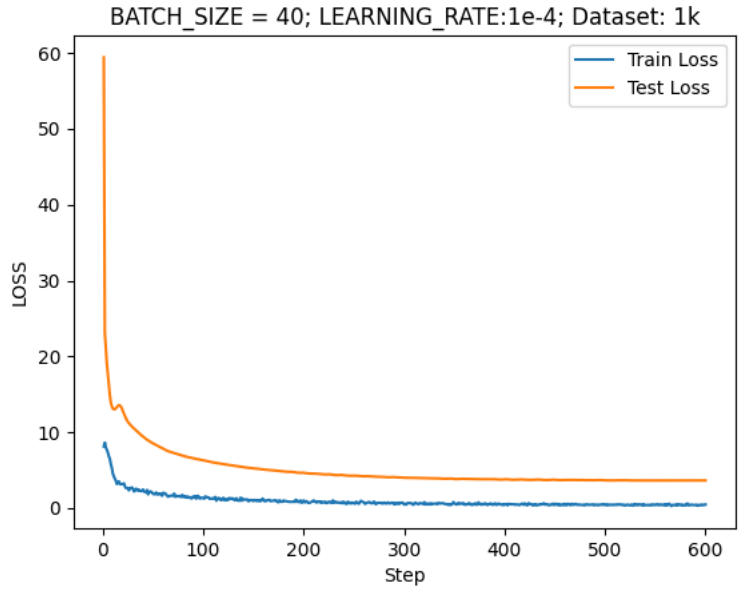
Origin Prompt	CLIP predict	ResNet predict
portrait of a cloaked female demoness, evil, ominous, luscious, ginger hair, stunning, detailed, by artgerm, by greg rutkowski, by luis royo, by pixar, by myazaki, gothic, final fantasy, fantasy, medieval	portrait of a beautiful woman with horns, large piercing blue eyes, long wavy black hair, long black dress with silver jewels, black bat wings on back, detailed background, by tom bagshaw!	a 9 of a of vintage, wow detail, intricate environment ~ ones golden hourada, houron english and ei wearing headphones negative nope no but it's not monaisaisa wearing headphones she mon not

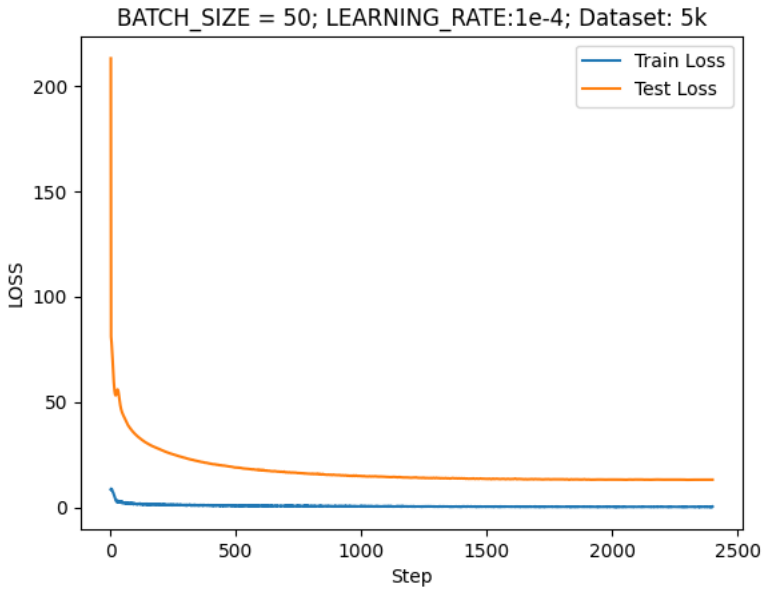
Origin Prompt	CLIP predict	ResNet predict
portrait of a dancing eagle woman, beautiful blonde haired lakota sioux goddess, intricate, highly detailed art by james jean, ray tracing, digital painting, artstation, concept art, smooth, sharp focus, illustration, artgerm and greg rutkowski and alphonse mucha, vladimir kush, giger 8 k	portrait of a dancing eagle woman, beautiful blonde haired lakota sioux goddess, intricate, highly detailed, ray tracing, digital painting, artstation, concept art, smooth, sharp focus, illustration, art by artgerm and g.	arait of a, r ,ism, 4, octane atherine ones greasley vic wearing headphones negative nope no not no not no not but's's not mona lisa is wearing headphones she mon not notaa

Origin Prompt	CLIP predict	ResNet predict
designed by giorgetto giugiaro stylized poser of a single 1 9 7 7 pontiac firebird 1 9 5 8 plymouth fury ((mclaren f 1)) delorean, sleek neon lights, ektachrome photograph, volumetric lighting, f 8 aperture, cinematic eastman 5 3 8 4 film	A portrait of james tissot	arait of a, single, rut , 4, light, 7, jeaneery atherine ones grey vehicle

Origin Prompt	CLIP predict	ResNet predict
complex 3 d render hyper detailed ultra sharp futuristic beautiful biomechanical humanoid woman with porcelain ivory face, medium shot portrait, close - up, filigree lace, iris van herpen cyberpunk daisies corals haute couture headdress with rhizomorph finials spires, brackets, fractal embroidered puffballs, octane render, 8 k	ultra sharp, medium shot portrait, brackets, octane render, 8 k, beautiful	a 3 of render hyper detailed ultra sharp beautiful, 8 k, futurism 4 k, octane render , ric estrada, ron english and ei

• Output analysis



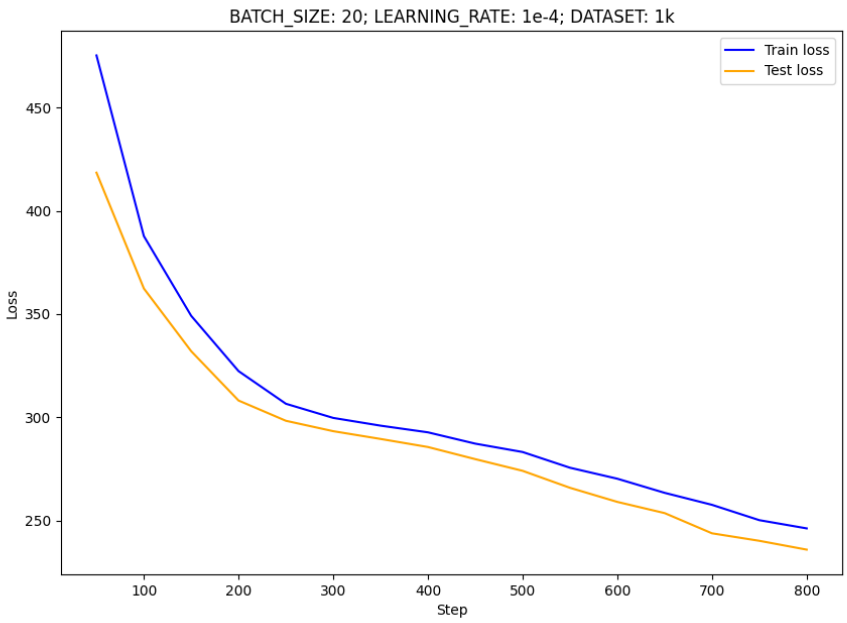


Text similarity	Loss	Training speed
——	10	90 s/epoch

The loss of the 1k data set dropped below 10, while the loss value of the 5k data set was higher.

When working with a dataset of 1,000 samples, increasing the batch size can be a viable approach if the loss value is substantial. However, when dealing with a larger dataset of 5,000 samples, this strategy might be limited by computer performance constraints. As a result, the loss may be higher compared to the 1,000-sample dataset scenario.

ResNet Model



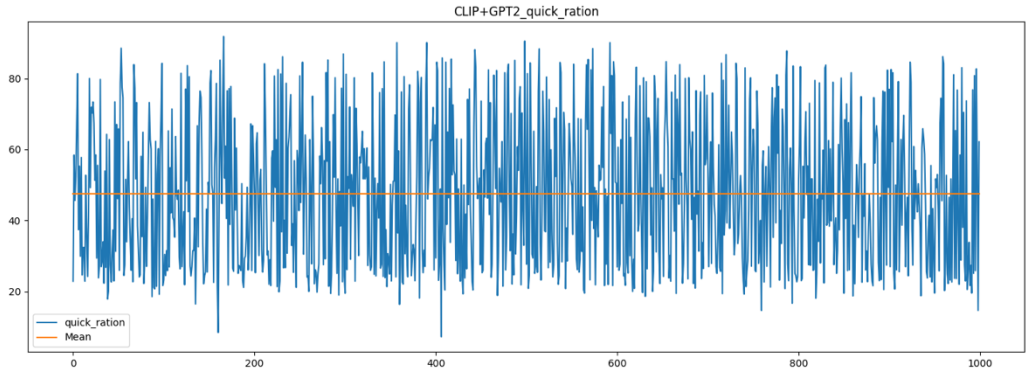
Text similarity	Loss	Training speed
——	200	100 s/epoch

From the figures we know that the loss of CLIP model is much lower than that of ResNet and LSTM.

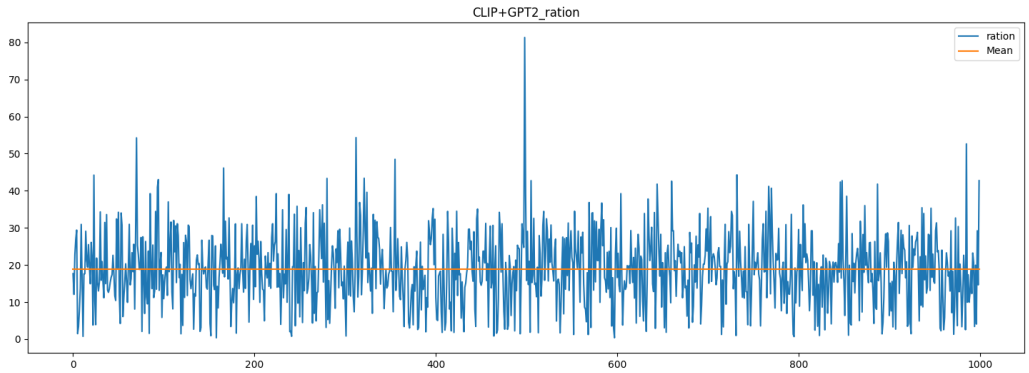
CLIP is efficient model which runs faster and consumes less resource compared to ResNet. We predicted that the clip model would perform better, and the results confirmed it.

Compared to the combination of ResNet and LSTM, CLIP potentially delivers superior performance for "image to prompt" tasks due to its more nuanced understanding of the relationship between images and text, developed during its training process. While the ResNet+LSTM combination is capable of handling such tasks, they process images and text separately. Even when they are connected through a fully connected layer, they may still fail to adequately capture the interplay between images and text. Conversely, CLIP is trained on

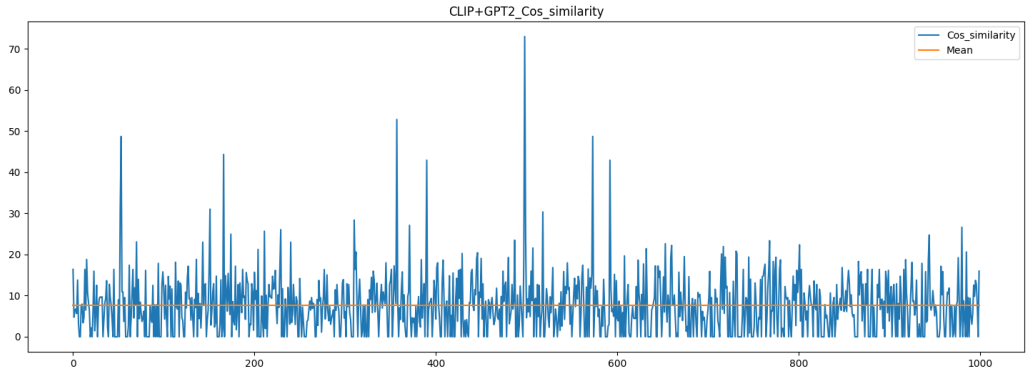
both images and text simultaneously, it is better to teach the model how to associate images with text.



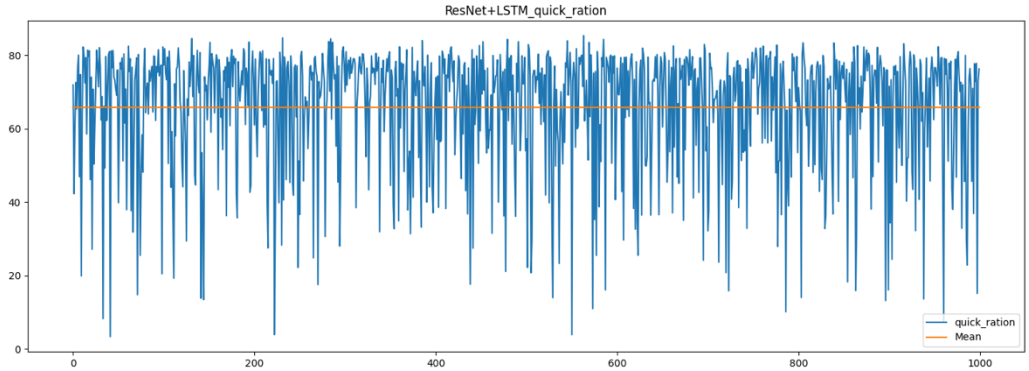
Model name	mean	variance	max
CLIP+GPT2_quick_ration	48%	427.25	100%



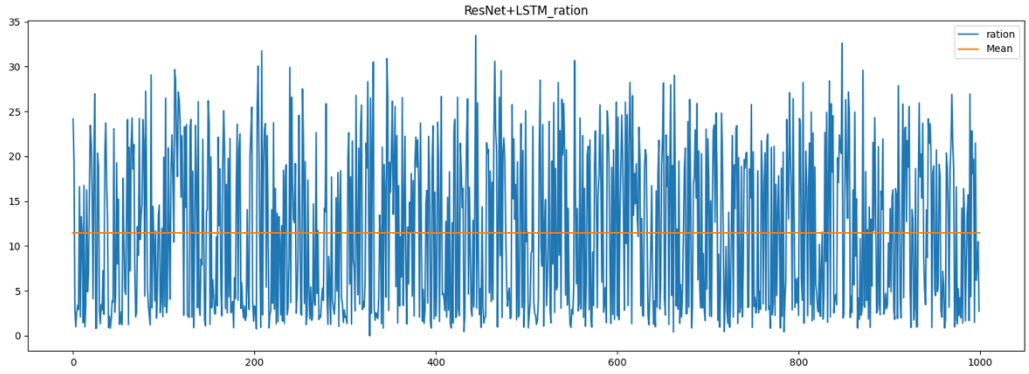
Model name	mean	variance	max
CLIP+GPT2_ration	19.5%	96.34	80%



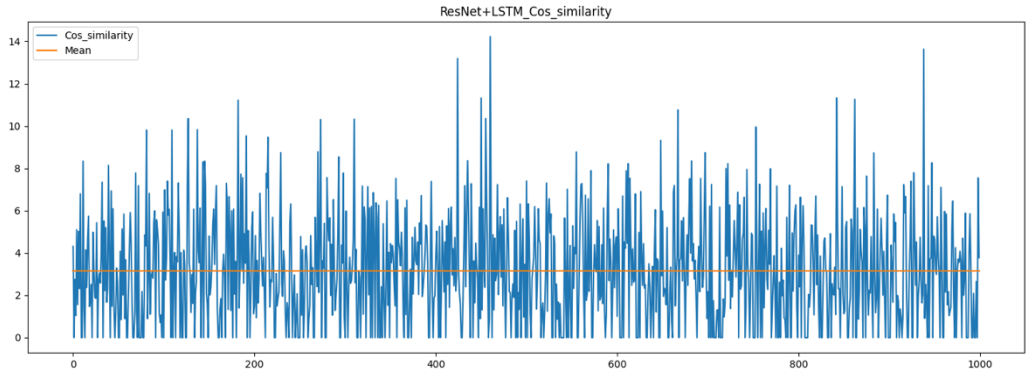
Model name	mean	variance	max
CLIP+GPT2_Cos_similarity	8.5%	52	70%



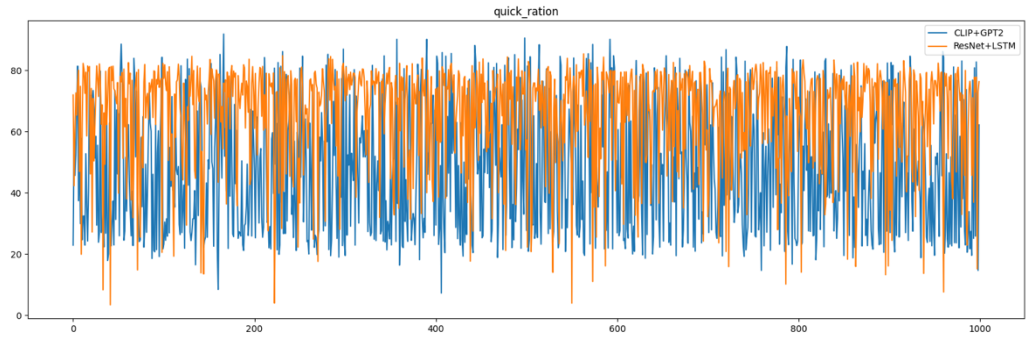
Model name	mean	variance	max
ResNet+LSTM_quick_ration	65%	280.86	85%

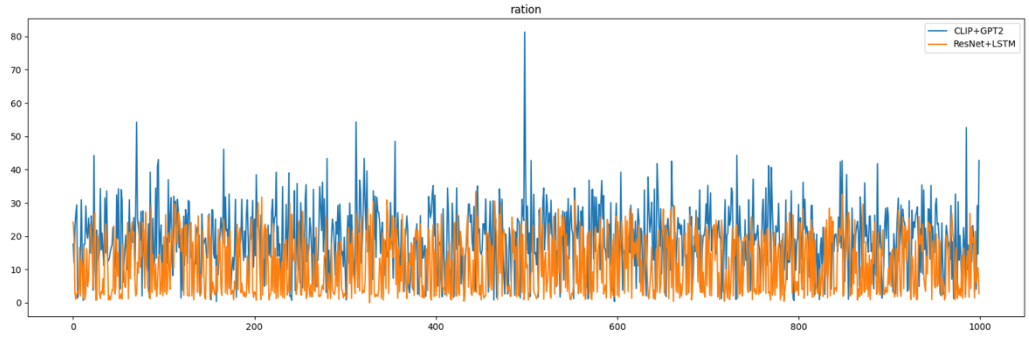


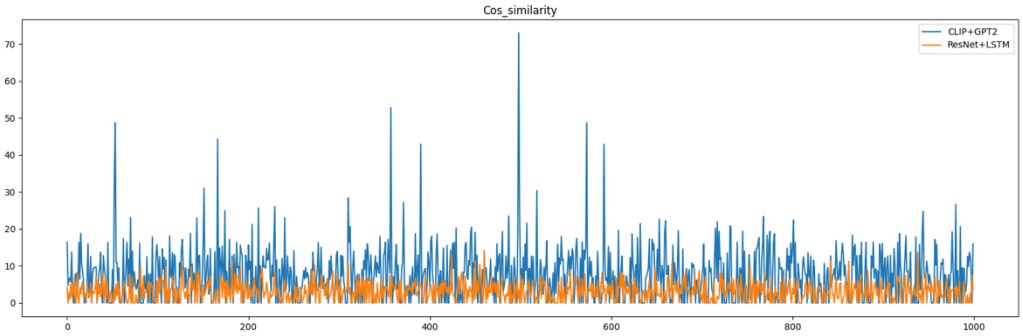
Model name	mean	variance	max
ResNet+LSTM_ration	13%	78.96	34%



Model name	mean	variance	max
ResNet+LSTM_Cos_similarity	3%	6.7	14%







The figures show that CLIP+GPT2 model has better performance while ResNet model is stabler and have lower variance.

High variance indicates that the model fits the training data too closely, which may lead to poor performance on new data. Low variance shows that the model fits the training data relatively loosely, but it can also lead to poor performance on the training data, that is, underfitting. The reason for the large variance may be that the data set is too small, because the larger the amount of data, the easier it is able to learn a wider range of patterns. Another reason for the higher variance of the clip model may be its higher complexity than ResNet. More complex models can better fit to training data, but also prone to overfitting, resulting in higher variance. If evaluate the models through quick ratio, ResNet is better. However, CLIP got better performance on cosine similarity and ratio.

Model Complexity: ResNet and LSTM, in their own rights, are complex models designed to address distinct types of data – namely images and sequential data. The fusion of these two models might contribute to an augmentation in the overall model complexity. As a consequence, this could potentially add intricacies to the training process. This challenge might be amplified when encountering high-dimensional or large-scale datasets, where the likelihood of experiencing complications during training could increase.

Why ResNet and LSTM perform not good than CLIP+GPT2?

- Vanishing/Exploding Gradient Problem: LSTM is crafted with the intention to mitigate the vanishing gradient issue that is commonly associated with traditional RNN models. However, in the landscape of practical applications, the challenges of vanishing or exploding gradients might persist occasionally, particularly in the context of deep networks.
- Model Capacity: The capacity of the model might be a delicate balance to strike. There could be instances where a model of relative simplicity might find it challenging to

fully encapsulate the complexity of the data, potentially leading to diminished returns from optimization efforts in terms of loss reduction. Alternatively, a highly intricate model might veer towards overfitting, which could potentially impede significant loss reduction on validation or test sets.

Part V

Summary

Summary

This project is the inverse of prompt word generation, that is, using images to generate prompt words.We implemented ResNet+LSTM model as well as CLIP+GPT2 model to learn the combination of computer vision models with natural language models as well as understand how pictures are predicted to be cue words.

Achievements

Despite the constraints of limited testing time and computational power, our predictions closely mirrored the actual values for a number of images, accurately capturing the core content depicted in these pictures. Intriguingly, for some image predictions, we achieved an exact match with the actual values for the initial part of the description. For instance, our prediction – 'portrait of a dancing eagle woman, beautiful blonde haired Lakota Sioux goddess, intricate, highly detailed, ray tracing, digital painting, ArtStation, concept art, smooth, sharp focus, illustration, art by Artgerm and g.' – perfectly aligned with the first half of the actual description – 'portrait of a dancing eagle woman, beautiful blonde haired Lakota Sioux goddess, intricate, highly detailed art by James Jean, ray tracing, digital painting, ArtStation, concept art, smooth, sharp focus, illustration, Artgerm and Greg Rutkowski and Alphonse Mucha, Vladimir Kush, Giger 8 k.

Challenge

When there are too many or too few samples of specific categories, the model may be biased towards predicting the category with more samples, leading to poorer recognition performance for minority categories. This makes it easy for the model to misidentify men as women.

Due to the limitations of local devices, we used the 2060 and 3050 graphics cards and M1 for preliminary training, and after success, we used the GPU P100 on Kaggle for training. Due to Kaggle's restrictions, we were only allowed to test for 30 hours at a time, preventing further model training. At the same time, we used pickle for compression to avoid the CPU memory limit of Kaggle when directly calling the Diffusiondb.

We attempted to implement this project using machine learning methods such as regression classification and SVM. However, we found that traditional machine learning has

significant disadvantages in image-to-text conversion. First, image data is usually high-dimensional, with each pixel point being considered a feature, often leading to the "curse of dimensionality" for traditional machine learning algorithms. Secondly, the image's relationship and corresponding text prompt are usually complex and non-linear. Traditional machine learning methods often perform poorly in dealing with such non-linear relationships. We tried to give each image classification and label each block of each image, manually performing feature engineering to extract meaningful features from the images, but this is almost impossible for about 10k datasets. We attempted a test with a 1k dataset, but the results were unsatisfactory. Since the dataset contained fewer fields, many of the output results had no relation to the formal prompt. We found that traditional machine learning algorithms often need help to meet training efficiency and effectiveness requirements when facing large amounts of data.

For the choice of model, most of the team members were unfamiliar with neural networks. We read many materials and discussed the advantages and disadvantages of neural network models such as VGG, ResNet, Transformer, LSTM, ViT, and CLIP, as well as language models such as GPT-2 and BERT. Ultimately, we implemented the model based on ViT's CLIP, ResNet and LSTM.

Feelings

This project is a combination of computer vision and natural language modeling, which is full of challenges and difficulties. We implemented the relevant models and I think we were very successful! [Shengyu Chen (George)]

As a machine learning beginner, this project is a good opportunity to learn this field. For different types of tasks, we need to learn to use different machine learning methods in order to achieve higher performance. [Enjie Zhang]

I was on the team, the only undergraduate. I was still apprehensive about working with graduate students and worried that I wouldn't be able to keep up with them, but luckily everyone was very patient. We finished perfectly. [Jinghan Wang (William)]

Firstly it was a great opportunity for me to learn machine learning with others. I am a very introverted person who hardly talks to strangers in my life and spend my days travelling

between classroom, library and home. Coupled with the fact that I was not exposed to machine learning as an undergraduate, I have to admit that 9417 was a bit more challenging for me, the first two Assignments were very crushing and difficult for me to cope with, coupled with the fact that I'm not that smart. So much so that I searched for a long time to find a suitable teammate. In the end I decided to set up a team on my own and managed to find teammates, but the results were not that satisfactory, two people, halfway through, dropped out and it did have a considerable impact on the progress and everyone's confidence. We also had some arguments in the middle of the meeting, which really challenged me a lot, not only the academic pressure, but also the need to deal with the relationship with my classmates, and let me feel what it is like to really hand over a project with my “colleagues". [Xianzhe zhang (Scott)]

future outlook

While our accuracy still requires improvement, we are confident to generate more accurate prompts for simpler images. As shown in table 1, some results are highly similar to the original text. Welcome to boost our project! [4]

We have some ideas to improve the performance of models for such stable diffusion tasks in future works.

1. Further process prompts given by original datasets. Some words like ‘4k’, ‘8k’ which assumed to stand for the resolution of raw images can be removed to reduce noise during training, because they do not contribute to the meaning of images. As mentioned above, content that requires background knowledge rather than a glance, such as year numbers should not appear in training data as well.

2. Increase the data set. Due to time and computer performance constraints, we are currently only using 10,000 images. Users with higher scores on kaggle have used hundreds of thousands or even millions of images for training. Compared with theirs, our dataset is much smaller.

3. For this project, BLIP and BLIP-2 which born in the beginning of this year were better choices. BLIP-2 uses a frozen pre-trained image encoder and a large language model (LLM), training a lightweight 12-layer Transformer encoder between them, which has faster speed and smaller CPU occupancy than CLIP. Originally, BLIP was chosen, but due to the dropout of one team member, this new model were not be implemented and tested.

[4]https://github.com/GATA-Chen/COMP9417_Project