Text-to-Image Diffusion Models with Enhanced Semantic Understanding

CS726 Project

Team Sarovar

Vanilla Diffusion Models don't do so well at Semantic Understanding and Reasoning

- Text-2-Image tasks not only require understanding the semantics of the text but also figuring out the implicit information and knowledge grounded in text
- Often involving visual question answering: **counting**, **color** and **action**

(a) A collection of seven vintage glass bottles in different shapes and sizes, arranged on a windowsill

(b) Five dogs

Project Objectives

- Explore and experiment with possible research papers and implement approaches improving semantic understanding of a stable diffusion pipeline and compare the results with a vanilla implementation.
- Propose architectural changes and possible modifications to these implementations and attempt at further improving the semantic understanding

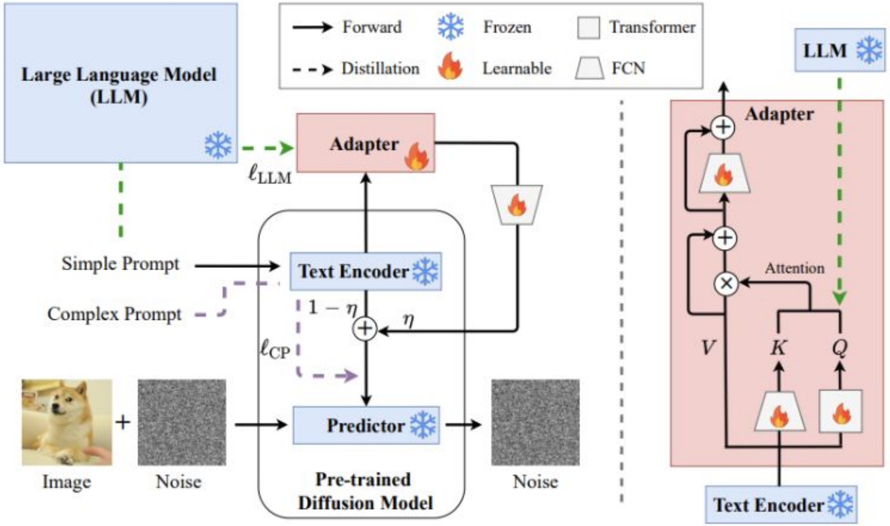
Explored Research Papers

- **SUR-adapter: Enhancing Text-to-Image Pre-trained Diffusion Models with Large Language Models (Zhong et. al)** : They propose simple-yet-effective parameter-efficient fine-tuning approach using an adapter for pretrained diffusion models
- **ELLA: Equip Diffusion Models with LLM for Enhanced Semantic Alignment (Hu et. al)** : They introduce an LLM adapter to equip pre trained text-to-image diffusion models with powerful LLMs to enhance image quality and timestep conditioning

SUR Adapter Architecture

The Architecture consists of two trainable neural networks g_{Ada}, g and with parameters ϕ_1 and ϕ_2

1. Adapter
   - **Adapter**:
$$g_{\text{Ada}}(f_{\text{En}}(p_{is});\varphi_2) = V_i' + V_i + h_1[V_i' + V_i],$$
where $V_i = f_{\text{En}}(p_s^i)$ and $V_i' = V_i \otimes \text{att}_i$.
   - The output of the adapter is transformed using $g(\cdot;\varphi_1)$, therefore:
$$g(g_{\text{Ada}}(f_{\text{En}}(p_{is});\varphi_2);\varphi_1) = g(V_i' + V_i + h_1[V_i' + V_i];\varphi_1),$$
   - Semantic information input to the predictor:
$$c'_{\text{LLM}} = \eta\, c_{\text{LLM}} + (1-\eta)\cdot f_{\text{En}}(p_s^i).$$
2. - Final Loss function is given by:
$$l_{\text{total}}(\phi) = \lambda_1 \cdot l_{\text{LLM}}(\phi) + \lambda_2 \cdot l_{\text{CP}}(\phi) + l_t^{\text{simple}}(\phi)$$
   - The intermediate loss expressions areas follows:
$$\ell_{\text{CP}}(\phi) = \text{KL}\left(\frac{c'_{\text{LLM}}}{\tau}, \frac{f_{\text{En}}(p_c^i)}{\tau}\right)$$
$$\ell_{\text{LLM}}(\phi) = \text{KL}\left[\frac{W_0 f_{\text{LLM}}(p_{is})}{\tau}, \frac{Q_i}{\tau}\right]$$



SUR Adapter (Experimentation)

- The default codebase provided at https://github.com/Qrange-group/SUR-adapter was not usable since the knowledge representation from LLAMA 13B of the input prompts were removed from public due to copyright restrictions
- We first attempted to generate these knowledge representation vectors as per format suggested in the paper by LLAMA 13B inference of SURD dataset prompts and extraction of the 40th layer output

SUR Adapter (Experimentation)

```python
def generate_prompt2vec(prompt):
    inputs = tokenizer(prompt, return_tensors="pt")

    with torch.no_grad():
        outputs = model(**inputs, output_hidden_states=True)

    vec = outputs.hidden_states[39].mean(1).squeeze(0)
    return vec
```

```python
x = generate_prompt2vec('a colorful animal with big eyes on a blue background')
```

```python
print("vec:", x, "shape:", x.shape)
```

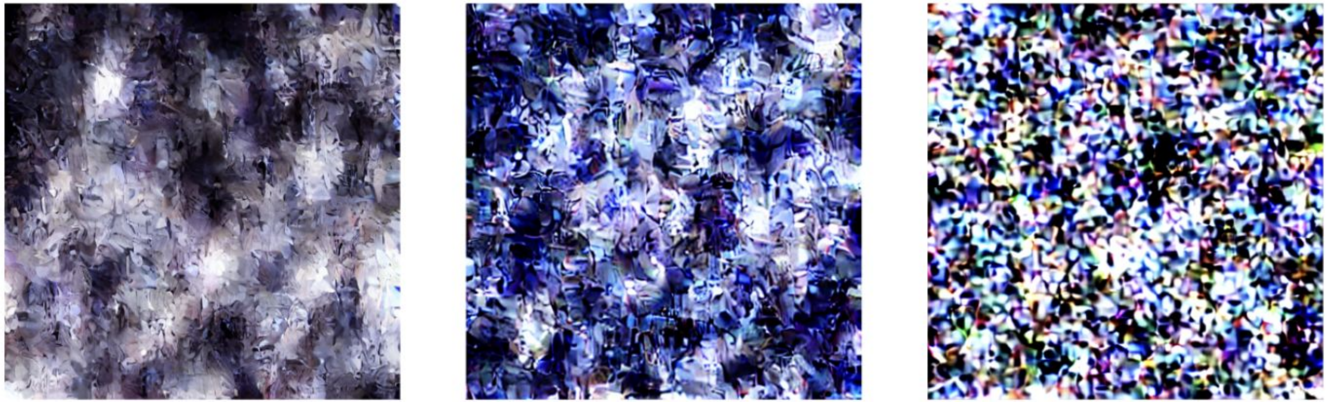
```
vec: tensor([1.8281, 0.3691, 0.8008,  ..., 0.5933, 1.0996, 0.8530],
       dtype=torch.float16) shape: torch.Size([5120])
```

```python
MODEL_NAME = "meta-llama/Llama-2-13b-hf"
```

```python
prompt_to_vec_dict = {}
for i in tqdm(train_dataset):
    curr_caption = i['caption_text']
    prompt_to_vec_dict[curr_caption] = generate_prompt2vec(curr_caption)
```

- Using this, we trained the SUR Adapter with a default value of η = 0.1, as suggested in the paper but the results obtained were completely bogus even after 5000 training steps

SUR Adapter (Experimentation)



(a) 1000 training steps (b) 2000 training steps (c) 5000 training steps

Figure 8: Generated images with varying training steps for the prompt: “An aristocratic maiden in medieval attire with a headdress of brilliant feathers”

- We think that this arises due to the fact that there is no instruction provided to the LLM about what it must do with the prompt

Drawbacks of SUR

1. Unrealistic Dataset

   Too complex prompts from simple prompts

2. Knowledge Distillation

   Linear Transform of LLM hidden state

3. Loss Function

   KL_DIVERGENCE

4. SUR influence η: Not significant

SUR Adapter (Proposed Modifications)

- **Different Dataset**: With simple prompt, complex image and different theme

  Classic Datasets lead to learning identity transform

- **LLAMA based Prompt Enrichment (**Later)
- **Usage of Cosine Similarity for Embedding Alignment**: The paper uses KL divergence based loss for isn't quite effective in capturing the similarity between text embeddings, so we use a cosine similarity based loss
- **Adaptive Improvement to Prompt**: We make the factor η learnable, adding an additional fully connected network (FCN) which takes as input the text embedding, to allow handling complex prompts, and at the same time provide significant influence from SUR

Dataset

image image · *width (px)*  272 1.28k	**text** string · *lengths*  33 103
	
	A cartoon character with a skeleton head and a white helmet.
	A cartoon character is holding a bone in its mouth.
	A cartoon character with a large foot and claws.
	A cartoon character is wearing a blue dress and boxing gloves.

LLAMA based Prompt Enrichment

- Implemented a prompt to LLAMA for providing: complex semantical and reasonable captions for simple captions
- We create a supervised (simple, complex) caption pairs in a dict which will be used during training

```python
def get_caption_detailing_prompt(caption):
    base_prompt = '''
    Please generate the long prompt version of the short one according to the given examples. Long prompt version should consist of 3 to 5 sentences. Long prompt version must specify the color, shape, texture or spatial relation of the included objects. DO NOT generate sentences that describe the surroundings of the object!!!

        Short: A calico cat with eyes closed is perched upon a Mercedes.
        Long: a multicolored cat perched atop a shiny black car. the car is parked in front of a building with wooden walls and a green fence. the reflection of the car and the surrounding environment can be seen on the car's glossy surface.

        Short: A boys sitting on a chair holding a video game remote.
        Long: a young boy sitting on a chair, wearing a blue shirt and a baseball cap with the letter 'm'. he has a red medal around his neck and is holding a white game controller. behind him, there are two other individuals, one of whom is wearing a backpack. to the right of the boy, there's a blue trash bin with a sign that reads 'automatic party'.

        Short: A man is on the bank of the water fishing.
        Long: a serene waterscape where a person, dressed in a blue jacket and a red beanie, stands in shallow waters, fishing with a long rod. the calm waters are dotted with several sailboats anchored at a distance, and a mountain range can be seen in the background under a cloudy sky.

        Short: A kitchen with a cluttered counter and wooden cabinets.
        Long: a well-lit kitchen with wooden cabinets, a black and white checkered floor, and a refrigerator adorned with a floral decal on its side. the kitchen countertop holds various items, including a coffee maker, jars, and fruits.

        Short:
        %s
    '''
    return base_prompt % caption
```

LLAMA based Prompt Enrichment

```
unet_loss = F.mse_loss(
    model_pred.float(), target.float(), reduction="mean"
)

loss = unet_loss + args.llm_loss_weight * llm_loss
```

```
def get_detailed_caption_with_llama(caption):
    t1 = time.perf_counter()
    sequences = pipeline(get_caption_detailing_prompt(caption))
    if DEBUG: print("llama inference took:", time.perf_counter() - t1)
    return sequences[0]['generated_text'].strip(" \nLong:").split("\n")[0]
```

```
for elem in batch['captions']:
    complex_prompt.append(llama_prompts[elem])

complex_tokens = clip_tokenizer(
    complex_prompt,
    return_tensors='pt',
    max_length=clip_tokenizer.model_max_length,
    padding="max_length",
    truncation=True
)
complex_tokens_ids = complex_tokens.input_ids
complex_tokens_ids = complex_tokens.input_ids.to(accelerator.device)

llama_embeddings = clip_text_encoder(complex_tokens_ids, return_dict=False)[0]

simple_tokens = clip_tokenizer(
    batch["captions"],
    return_tensors='pt',
    max_length=clip_tokenizer.model_max_length,
    padding="max_length",
    truncation=True
)
simple_tokens_ids = simple_tokens.input_ids
simple_tokens_ids = simple_tokens.input_ids.to(accelerator.device)

simple_embed = clip_text_encoder(simple_tokens_ids, return_dict=False)[0]
out, _ , _ = suradapter(simple_embed)
```

```
    "A cartoon character with a muscular arm and a frowning f
ace.": "a cartoon character with a robust physique, standing
with his left arm bent and his right hand on his hip. he has
a stern expression on his face, and his bright yellow skin co
ntrasts with his dark blue shorts and red sneakers. the backg
round is a bright and colorful cityscape with skyscrapers and
billboards.",
    "A cartoon monster with a pink nose and a frown on its fa
ce.": "a brightly colored monster with a large pink nose and
a frown on its face, perched on top of a fluffy white cloud.
the cloud is surrounded by a clear blue sky with a few white
clouds floating lazily by.",
    "A cartoon drawing of a turtle with a mouth open.": "a vi
brant cartoon of a turtle with a wide-open mouth, its teeth v
isible in the shadows. the turtle's shell glistens in the lig
ht, and its eyes are large and expressive, as if about to mak
e a joke. the background is a bright blue sky with fluffy whi
te clouds.",
```

Cosine Similarity Loss

Intuition:

1. Averaging along embedding dimension results in finding the mean word
2. Mean word: point in embedding dimension that represents the overall meaning of sentence

```python
def Cos_loss(e1, e2):
    e1_avg = torch.mean(e1, dim=1)
    e2_avg = torch.mean(e2, dim=1)


    # Calculate cosine similarity loss
    sim_loss = 1-cosine_similarity(e1_avg, e2_avg)
    return sim_loss
```

```python
unet_loss = F.mse_loss(
    model_pred.float(), target.float(), reduction="mean"
)

loss = unet_loss + args.llm_loss_weight * llm_loss
```

Modified Cosine Similarity Loss

Text 1	Text 2	Cosine Loss	Distillation Loss
Hello! How are you?	Hi, How your doing	0.144	147.963
Hello! How are you?	Bye! Mom where are you going	0.501	183.568
Hello! How are you?	White tiger and blue elephant	0.644	83.317
Hello! How are you?	Horse on the moon	0.603	68.026
Hello! How are you?	Donkey in Mars	0.608	72.141
Hello! How are you?	Violet lion and green donkey	0.646	81.022
Hello! How are you?	Purple cheetah and yellow hippo	0.658	87.245
Hi, How your doing	Bye! Mom where are you going	0.518	41.167
Hi, How your doing	White tiger and blue elephant	0.666	145.305
Hi, How your doing	Horse on the moon	0.617	123.277
Hi, How your doing	Donkey in Mars	0.625	115.777
Hi, How your doing	Violet lion and green donkey	0.684	151.838
Hi, How your doing	Purple cheetah and yellow hippo	0.668	145.109

Sample Results

(a) "A golden sun setting over a calm ocean, with orange and pink hues appearing in the sky"

(b) "A gymnast performing a balance beam routine with graceful flips and twists"

(c) "A skateboarder doing a kickflip over a set of stairs"

(d) Pikachu by SUR (New Loss)

SUR Adapter : Results

(a) "Three fluffy white kittens playing with a ball of yarn on a bright green carpet"

(b) SUR Adapter Results

(c) "A collection of seven vintage glass bottles in different shapes and sizes, arranged on a windowsill"

(d) SUR Adapter Results

(e) "An aristocratic maiden in medieval attire with a headdress of brilliant feathers"

(f) "An aristocratic maiden in medieval attire with a headdress of brilliant feathers"

6.1 Color Analysis

(a) Clip Stable Diffusion Model

(b) SUR Adapter 3000 steps

(c) SUR Adapter 4000 steps

(d) SUR Adapter 5000 steps

Figure 9: Analysis of results for the prompt: "A vibrant butterfly with iridescent wings in shades of blue, green, and purple, perched on a bright pink flower"

6.2 Action Category Analysis

(a) Best Picture For Clip Stable Diffusion Model

(b) SUR Adapter 3000 steps

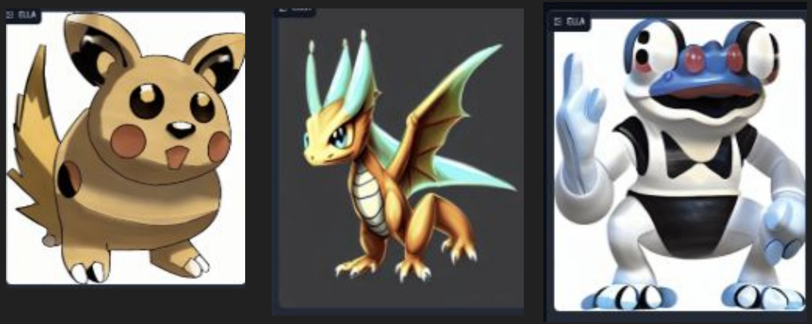
(c) SUR Adapter 4000 steps

Figure 11: Analysis of results for the prompt: "A chef tossing a pizza dough in the air in a kitchen"

Pokemons for Fun!

Figure 15: Analysis on Pokemon prompts

ELLA (Next…)



ELLA Adapter

1. Using powerful LLMs as text encoders:

   T5, LLama, or Mistral

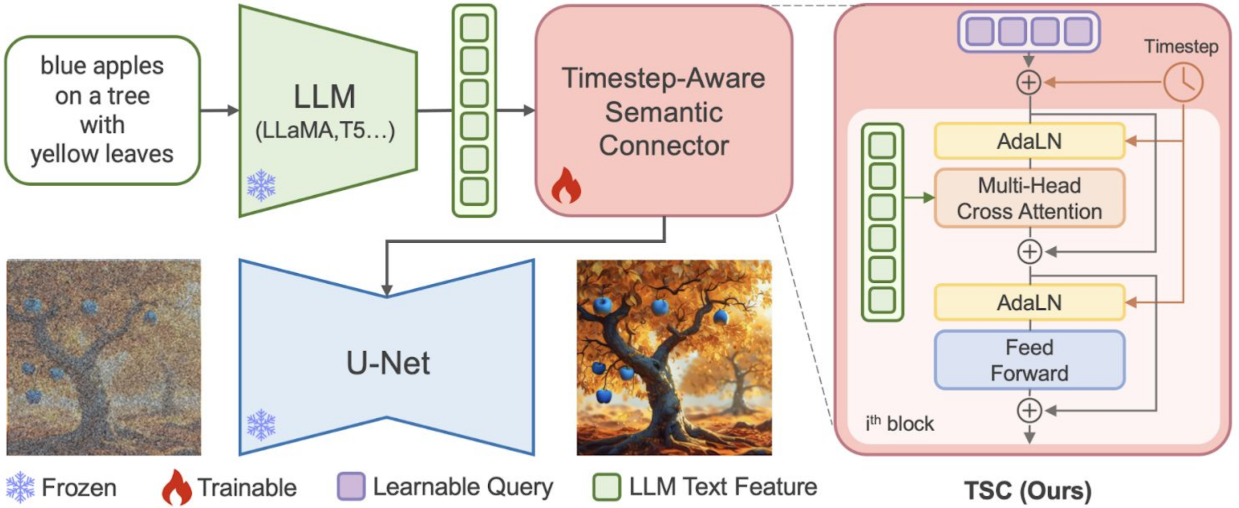
2. Developing adapters that are timestep aware:

   Low Frequency Features: Background

   High Frequency Features: Facial Details and Fine Details

3. All this without changing the conditional U-Net

Architecture



The overview of ELLA.

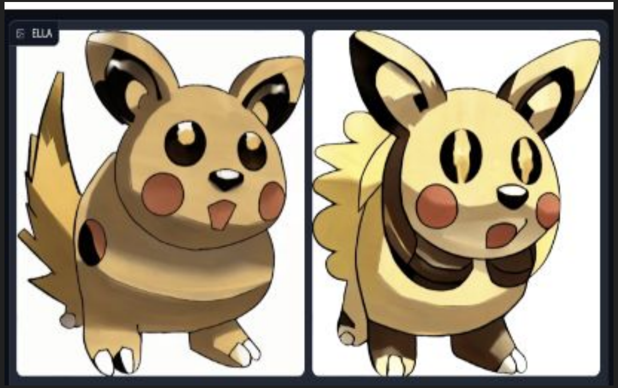
Our Trained ELLA Results

An aristocratic girl in medieval finery and a headdress of bright feathers drinking afternoon tea

Three fluffy white kittens playing with a ball of yarn on a bright green carpet


ELLA

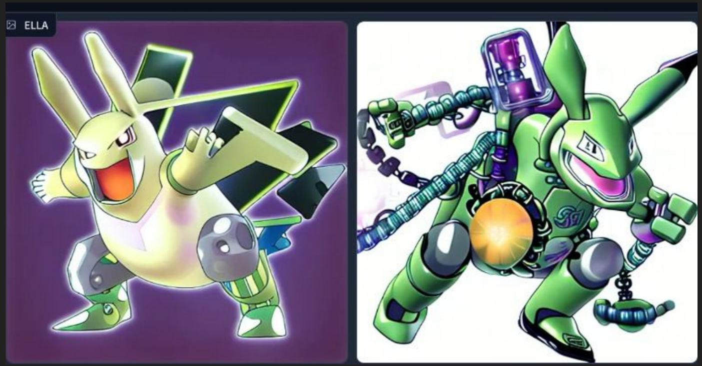

ELLA


ELLA


ELLA


ELLA


ELLA


ELLA