PikaPikaGen

Generative Synthesis of Pokémon Sprites from Textual Descriptions

Student: Luigi Battista
Polytechnic University of Bari
l.battista4@studenti.poliba.it

Teacher: Prof. Vito Walter Anelli
Deep Learning – Project 2025_VI

Contents

Project Overview: Text-to-Image Synthesis

The assigned project involves developing a model capable of synthesizing **Pokémon-style sprite images** from their corresponding **Pokédex descriptions**. Specifically, the task required designing an Encoder–Decoder architecture with an Attention mechanism to bridge the textual and visual modalities.
To address this, a **conditional Generative Adversarial Network (cGAN)** framework was adopted: the generator acts as a decoder conditioned on encoded text, fulfilling the Encoder–Decoder design since only the generator is retained at inference time.

Text-to-Pokémon generation Task Input
This Pokémon uses its psychic while it sleeps. The contents of dreams affect the powers that Pokémon wields

Dataset

The original dataset consisted of a **single** high-quality segmented image per Pokémon and a short textual description – often **lacking crucial visual details** such as color, shape, body structure and distinctive features. This made it inadequate for training a robust text-to-image generative model.
To overcome these limitations, a **custom dataset** was built from scratch to improve both **visual diversity** and **linguistic variation**. This required the design of a multi-step data collection and curation pipeline:

1. **Image Collection** – Gathering multiple anime-style images per Pokémon from various online sources.
2. **Image Segmentation** – Removing backgrounds to focus on character-specific visual content.
3. **Description Curation** – Writing detailed, visually grounded captions for each Pokémon.
4. **Paraphrasing** – Generating multiple textual variants to improve language robustness and diversity.

Dataset: Image Collection

The first step in dataset creation involved **collecting** a large and diverse set of Pokémon images. Given the lack of any sufficiently varied dataset online suitable for generative modeling, I developed an automated pipeline to gather anime-style illustrations, focusing on first-generation Pokémon due to their visual popularity (115 Pokémon).

- **Image Retrieval:** performed using the **icrawler** Python library, targeting both *Google* and *Bing*. To enhance relevance, search queries combined the Pokémon name with keywords such as *official anime art*, *anime screenshot* and *anime official*, retrieving up to 300 images per Pokémon.
- **Deduplication:** Applied **perceptual hashing** (`phash`) via the `imagehash` library. This method effectively identified and removed visually similar images, even if resized or slightly edited.

Despite filtering efforts, many irrelevant images remained–such as merchandise photos, pixel art, or inconsistent fan art– necessitating a final manual cleaning phase to ensure visual consistency.

Dataset: Image Segmentation

After collecting the images, the next step involved **removing backgrounds** to isolate each Pokémon character. This helped the model focus purely on the character's appearance without being influenced by distracting or inconsistent backgrounds.

Initial tests with top background removal models from Hugging Face – such as *RMBG-1.4* and *BiRefNet* – produced unsatisfactory results, often missing body parts or retaining background noise.

For higher quality, I adopted the **Grounded-Segment-Anything (GSA)** pipeline, which combines:

- **Grounding DINO**: detects object regions from a text prompt (e.g., "Pokémon character") in a zero-shot manner
- **Segment Anything Model (SAM)**: generates high-quality masks from the bounding boxes

For each image, up to **5 candidate masks** were generated. If an image already had transparency, it was preserved. The best mask per image was selected manually to ensure quality. This process produced **about 40 clean, background-free images** per Pokémon.

Dataset: Description Curation

To guide the text-to-image model, I curated a set of **concise, visually-focused descriptions** for each Pokémon. The starting point was the *Pokémon Dataset 2024* from Kaggle, which aggregates character details from *Bulbapedia*, a fan-maintained Pokémon encyclopedia.

While these entries are comprehensive, they often include irrelevant details such as evolutionary traits, behavioral lore, and competitive roles. To focus purely on visual characteristics, I reprocessed the entries using **GPT-4o**, applying strict formatting rules.

Each description was:

- Limited to **40–60 words**, ensuring brevity and consistency
- Required to mention visual traits such as *shape, size, color, limbs, eyes, ears, tails* and any notable features
- Explicitly filtered to exclude non-visual content (e.g., battle roles, evolutions)

The result was a clean, uniform set of visual captions that clearly defined appearance cues.

Dataset: Paraphrasing

To improve the model's ability to generalize beyond fixed phrasings, I performed an additional **paraphrasing step**. This served two purposes: (1) enrich training by introducing phrasing variability, and (2) test the model's robustness on unseen paraphrases.

I evaluated top Hugging Face models for the paraphrasing task:

- *pegasus_paraphrase*
- *parrot_paraphraser_on_T5*
- *chatgpt_paraphraser_on_T5_base*

After comparison, `chatgpt_paraphraser_on_T5_base` emerged as the most reliable, producing high-quality paraphrases that retained the intended semantics.

For each description, I generated two paraphrased variants:

- `train_paraphrase`: Used in training, increasing phrasing diversity
- `test_paraphrase`: Reserved for testing, to evaluate generalization

Generation was guided by *beam search*, diversity penalties and *n-gram blocking* to maximize output quality and ensure uniqueness.

Dataset Splitting & Preprocessing

Image dataset was split into:

- **Train (90%)** – 4022 images
- **Validation (3%)** – 101 images
- **Intra-Class Test (7%)** – 403 images (same classes, unseen images)
- **Novel-Class Test** – 402 images (from unseen Pokémon species)

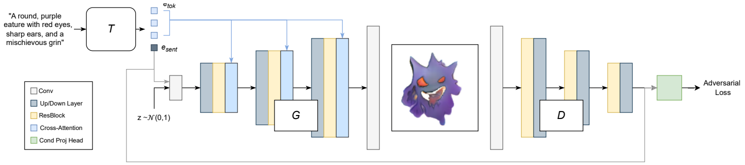
All images were preprocessed: converted to RGB (white background), saved as high-quality `.jpg`, resized and normalized to $[-1, 1]$.

Augmentation was applied only to the training set, using:

- Horizontal flipping
- Mild affine transformations (rotation, scale, translation)
- Subtle color jitter (brightness, contrast, hue, saturation)

During training, each sample had an 80% chance of using the original description (`description`) and a 20% chance of using a paraphrased variant (`train_paraphrase`), promoting linguistic diversity. `test_paraphrase` is only used at test time for novel-class testing.

Model Overall Architecture



In the generator, text conditioning is applied via two complementary pathways:

- **Global Conditioning:** the sentence embedding e_{sent} is concatenated with the latent noise vector z, providing high-level semantic context from the very beginning.
- **Local Conditioning:** token-level embeddings e_{tok} are injected through *cross-attention layers* at multiple stages, enabling fine-grained alignment between the generated visual content and the descriptive text.

In the discriminator, only the sentence embedding e_{sent} is used, injected at the final stage via a *conditional projection head*.

Text Encoder: Overview

To enable flexible text conditioning, the system supports two alternatives:

- **BERT-Mini**: a lightweight encoder as per original assignment.
- **CLIP ViT-B/32**: explored to leverage vision-aligned semantic embeddings.

Since CLIP produces 512-dimensional embeddings, a linear projection was added to map its outputs to the model's 256-dimensional latent space.

CLIP (Contrastive Language–Image Pretraining) is a multimodal framework trained to align image and text embeddings via a contrastive loss on 400M image–caption pairs.

- CLIP is composed of two encoders – a vision encoder and a text encoder. **Only** the **text encoder** is used.
- Tokenization uses **Byte-Pair Encoding (BPE)** with a vocabulary of 49,408 tokens.
- The model processes up to 77 tokens. Sequences longer than the model's maximum length are truncated, while shorter ones are padded.

Text Encoder: CLIP Architecture

Input tokens are embedded and combined with learned positional embeddings. The resulting sequence passes through a 12-layer Transformer stack with:

- Multi-Head Self Attention (MHSA) with 8 heads (head dim = 64)
- Feed-Forward Networks (FFN) with QuickGELU activations
- Residual connections and *Layer normalization* are applied after both the sublayers

At the output, two types of embeddings are produced:

- **Token embeddings** $(T \times 512)$
- **Sentence embedding** (1×512), obtained by applying a linear projection to final `[EOS]` token state

Component	**Configuration**
Tokenizer	Byte-Pair Encoding (BPE)
Maximum Input Length	77
Vocabulary Size	49,408
Embedding Dimension	512
Transformer Layers	12
Attention Heads	8
Head Dimension	64
FFN Hidden Dim	2048
Activation	QuickGELU
Normalization	LayerNorm
Output Dimensions	Token: (77×512), Sentence: (1×512)

Fine-Tuning Strategy

Due to the limited dataset size, a full fine-tuning of large pretrained encoders like *CLIP* posed overfitting risks. Instead, a **partial fine-tuning strategy** was adopted, where only select components were updated while the majority of parameters remained frozen.

Fine-tuned components:

- **Token Embedding Layer:** Allows the model to adapt input representations to domain-specific vocabulary. Positional embeddings were kept frozen.
- **Final Transformer Block:** Enables modest structural adaptation in the final contextual representations while preserving lower-layer features.
- **LayerNorm Parameters:** All affine parameters (scale γ, shift β) were fine-tuned, enabling lightweight adaptation of activation statistics to the characteristics of the downstream task.
- **Sentence Projection Head:** The projection layer producing the global sentence embeddings (e.g., `text_projection` in CLIP) was fine-tuned.

The additional linear projection used to map CLIP embeddings to 256 dimensions was trained *from scratch*.

Generator (I)

The generator progressively synthesizes high-resolution images from a latent vector z, conditioned on both the **sentence embedding** e_{sent} and **token-level embeddings** e_{tok}.

Conditioning Strategy:

- **Global (Sentence):** $z \in \mathbb{R}^{128}$ and $e_{\text{sent}} \in \mathbb{R}^{256}$ are concatenated and projected to an initial 4×4 feature map.
- **Local (Tokens):** e_{tok} is injected via cross-attention layers across multiple spatial resolutions.

Architecture:

- Composed of repeated groups: *Upsample* $\rightarrow$ *Residual Block* $\rightarrow$ *Cross-Attention*.
  - **Upsampling:** Bilinear interpolation $+$ 3×3 convolution (or optional transposed conv). Reduces checkerboard artifacts.
  - **Residual Block:** Two 3×3 convs with GroupNorm (32 groups) and SiLU activations. Ensure stable gradient flow in deeper networks.
  - **Cross-Attention:** Injects e_{tok} using 4-head attention (head dim $= 64$). Uses SDPA for memory efficiency and speed-up training. Queries $=$ image features; Keys/Values $=$ token embeddings.
- Ends with a **RGB Head:** Final 3×3 conv $+$ GroupNorm $+$ SiLU $+$ `tanh` for normalized RGB output.

Generator (II)

Table: Generator configuration for 128×128 resolution

Group	**Kernel**	**Stride**	**Padding**	**Channels**	**Resolution**
Initial Projection	–	–	–	256	4×4
Upsample 1	3×3	2×2	1	256	8×8
Residual Block 1	3×3	1×1	1	256	8×8
Cross-Attention 1	–	–	–	256	8×8
Upsample 2	3×3	2×2	1	256	16×16
Residual Block 2	3×3	1×1	1	256	16×16
Cross-Attention 2	–	–	–	256	16×16
Upsample 3	3×3	2×2	1	128	32×32
Residual Block 3	3×3	1×1	1	128	32×32
Cross-Attention 3	–	–	–	128	32×32
Upsample 4	3×3	2×2	1	64	64×64
Residual Block 4	3×3	1×1	1	64	64×64
Cross-Attention 4	–	–	–	64	64×64
Upsample 5	3×3	2×2	1	32	128×128
Residual Block 5	3×3	1×1	1	32	128×128
Cross-Attention 5	–	–	–	32	128×128
Final RGB Head	3×3	1×1	1	3	128×128

Discriminator (I)

The discriminator assesses both **realism** and **text-image alignment** of generated samples.

Architecture:

- Starts with an initial 3×3 conv to project RGB input to 32 channels.
- Stack of 5 **residual blocks with downsampling**:
  - Each block: two 3×3 convolutions $+$ LeakyReLU $(\alpha = 0.2)$
  - Downsampling is *integrated* within the block on both main and skip paths.
  - Default downsampling: strided conv $(4 \times 4$, stride=2). Optional: AvgPool (2×2)
- **Spectral Normalization** is applied to all conv layers. Spectral Norm constrains the Lipschitz constant of each layer by normalizing its weight matrix via its largest singular value. This ensures smoother gradients and prevents the discriminator from becoming too sharp or overconfident, which is known to destabilize GAN training.

Discriminator (II)

- **Conditional Projection Head:**
  - After last block, get 4×4 feature map.
  - Sentence embedding $e_{\text{sent}} \in \mathbb{R}^{256}$ is broadcast and concatenated along channels.
  - Two convs:
    - 3×3 conv + LeakyReLU
    - 4×4 conv to compress to scalar output
  - Output: **real/fake + semantic alignment score**

Table: Discriminator architecture configuration for 128×128 input resolution.

Block	**Kernel Size**	**Stride**	**Padding**	**Output Channels**	**Resolution**
Initial Conv	3×3	1×1	1	32	128×128
ResBlockD 1 + Down	3×3	2×2	1	64	64×64
ResBlockD 2 + Down	3×3	2×2	1	128	32×32
ResBlockD 3 + Down	3×3	2×2	1	256	16×16
ResBlockD 4 + Down	3×3	2×2	1	256	8×8
ResBlockD 5 + Down	3×3	2×2	1	256	4×4
Conditional Projection Head	$3 \times 3 + 4 \times 4$	1×1	1 / 0	1	1×1

Loss Functions (I)

The model is trained via adversarial learning with a minimax objective:

$$\min_G \max_D \mathcal{L}(G, D)$$

The generator learns to create realistic, text-aligned images, while the discriminator distinguishes real vs generated and evaluates text consistency.

Discriminator Loss:
The discriminator receives three input pairs:

1. Real image + matching text
2. Generated image + matching text
3. Real image + mismatched text

Its loss is a sum of hinge terms plus a gradient penalty:

$$\mathcal{L}_D = \mathcal{L}_{\text{real}} + \frac{1}{2}\mathcal{L}_{\text{fake}} + \frac{1}{2}\mathcal{L}_{\text{mis}} + \lambda_{\text{MA-GP}} \cdot \mathcal{L}_{\text{MA-GP}}$$

Real images with matching text:

$$\mathcal{L}_{\text{real}} = \mathbb{E}_{(x, e_{\text{sent}})\sim P_r}\left[\max(0, 1 - D(x, e_{\text{sent}}))\right]$$

Loss Functions (II)

Fake images with matching text:

$$\mathcal{L}_{\text{fake}} = \mathbb{E}_{z,(e_{\text{tok}},e_{\text{sent}})}\left[\max(0, 1 + D(G(z, e_{\text{tok}}, e_{\text{sent}}), e_{\text{sent}}))\right]$$

Real images with mismatched text:

$$\mathcal{L}_{\text{mis}} = \mathbb{E}_{x\sim P_r, e_{\text{sent}}\sim P_{\text{mis}}}\left[\max(0, 1 + D(x, e_{\text{sent}}))\right]$$

Matching-Aware Gradient Penalty (MA-GP):

$$\mathcal{L}_{\text{MA-GP}} = \mathbb{E}_{(x,e_{\text{sent}})\sim P_r}\left[\|\nabla_{x,e_{\text{sent}}} D(x, e_{\text{sent}})\|_2^6\right]$$

where

$$\nabla_{x,e_{\text{sent}}} D(x, e_{\text{sent}}) = \left[\frac{\partial D}{\partial x}, \frac{\partial D}{\partial e_{\text{sent}}}\right]$$

Generator Loss:
The generator aims to fool the discriminator by generating images that receive high scores when evaluated against their corresponding text descriptions:

$$\mathcal{L}_G = -\mathbb{E}_{z,(e_{\text{tok}},e_{\text{sent}})}\left[D(G(z, e_{\text{tok}}, e_{\text{sent}}), e_{\text{sent}})\right]$$

Results: Training Details (I)

Overview:

- Details about the architectural and training choices made that culminated in the final configuration used.
- A total of 25 training runs were conducted with variations in resolution, learning rates, regularization strength, attention injection method and optimizer schedules.

Model Settings:

- **Text Encoder:** Switched from BERT-Mini to CLIP (ViT-B/32) for better semantic alignment.
- **Image Resolution:** 128×128 chosen over 256×256 for stability and faster convergence.
- **Upsampling:** Bilinear interpolation + convolution preferred over transposed convolutions to avoid artifacts.
- **Conditioning:** Fixed cross-attention injection favored over learnable strength parameter due to training stability.
- **Discriminator:** Spectral normalization applied; downsampling via strided convolutions for feature preservation.

Results: Training Details (II)

Optimization Settings:

- Adam optimizer with $\beta_1 = 0.0$, $\beta_2 = 0.9$.
- Learning rates: Generator 1×10^{-4}, Discriminator 4×10^{-4}, CLIP encoder 2×10^{-5} (fine-tuning).
- Fixed learning rates + manual early stopping
- MA-GP regularization set to $\lambda_{\text{MA-GP}} = 1.5$..
- Batch size = 64
- Automatic Mixed Precision (AMP) training: adversarial losses in `float16`, MA-GP in `float32` for numerical stability.

Summary of Hyperparameters:

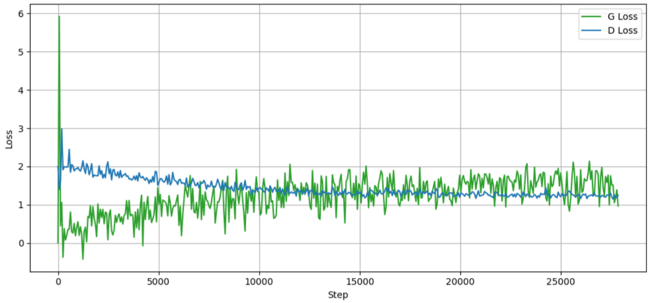
Component	**Value**
Resolution	128×128
Batch Size	64
Text Encoder	CLIP (ViT-B/32)
Conditioning	Cross-attention (fixed)
Upsampling	Bilinear + Conv
Downsampling	Strided Conv
Discriminator Norm.	Spectral Norm
Optimizer	Adam ($\beta_1 = 0.0$, $\beta_2 = 0.9$)
Gen. LR	1×10^{-4}
Disc. LR	4×10^{-4}
Text Enc. LR	2×10^{-5}
$\lambda_{\text{MA-GP}}$	1.5

Results: Training Monitoring (I)

Challenges in GAN Training:

- Maintaining balance between generator and discriminator is crucial.
- *Mode collapse*: generator outputs lack diversity, stuck on limited modes.
- *Training divergence*: unstable gradients cause loss spikes and failure to converge.
- Monitoring training dynamics is essential to detect and mitigate these failure modes.

Training Loss Curves: Initially, as discriminator improves, generator loss rises. Then both losses stabilize around epoch 240 (15,000 steps), indicating an adversarial equilibrium is reached.



Results: Training Monitoring (II)

Validation

- In GANs, traditional validation losses are *not* reliable indicators of generation quality or mode coverage.
- The GAN adversarial objective does not directly optimize for reconstruction or likelihood.
- As a result, validation loss cannot be used for early stopping or overfitting detection.

Validation Strategy Adopted:

- **Qualitative:** Visual inspection of generated images over training epochs to identify signs of instability or degradation in sample quality.
- **Quantitative:** Best checkpoint chosen based on lowest FID on held-out validation set.

Note: Training losses, qualitative samples and FID values were tracked and logged using TensorBoard.

Results: Evaluation Metrics (I)

Inception Score (IS) measures both the *quality* and *diversity* of generated images using a pretrained Inception v3 classifier.

For each generated image x, the classifier outputs a conditional class distribution $p(y|x)$. The marginal class distribution $p(y)$ is computed as the average over all generated images, i.e. $p(y) = \mathbb{E}_{x\sim p_g}[p(y|x)]$
The IS uses the Kullback–Leibler (KL) divergence to measure how much the prediction for each image differs from the overall average:

$$\text{IS}(G) = \exp\left(\mathbb{E}_{x\sim p_g}\left[D_{KL}\left(p(y|x) \,\|\, p(y)\right)\right]\right)$$

Intuition:

- $p(y|x)$ should be **peaky** (low entropy): the classifier is confident.
- $p(y)$ should be **diverse** (high entropy): the model covers many classes.
- **Higher IS = better**: confident individual predictions + diverse sample classes

Limitations:

- Does not compare with real data distribution
- Relies on alignment with ImageNet classes – poor fit for non-natural images.

Usage: Despite its early popularity, IS has been largely replaced by more reliable metrics like FID.

Results: Evaluation Metrics (II)

Fréchet Inception Distance (FID) evaluates how close the distribution of generated images is to that of real images in a learned feature space.

Both real and generated images are fed into a pretrained Inception v3 network. Features are extracted from an intermediate layer (the final average pooling layer), resulting in 2048-dimensional vectors.
FID computes the Fréchet distance (Wasserstein-2) between the empirical distributions of these features, modeled as multivariate Gaussians:

$$\text{FID}(X_r, X_g) = \|\mu_r - \mu_g\|_2^2 + \operatorname{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\right)$$

- μ_r, Σ_r: mean and covariance of real image features
- μ_g, Σ_g: mean and covariance of generated image features

Intuition:

- First term: content similarity (mean feature alignment)
- Second term: diversity similarity (covariance structure)
- **Lower FID = better**: closer match to real images in both content and variation

Limitations:

- Assumes Gaussian feature distribution — may not strictly hold
- Sensitive to sample size (e.g., usually 10k+ samples needed)

Results: Evaluation Metrics (III)

Kernel Inception Distance (KID) is a metric designed to reliably evaluate generative models, especially when **test set size is small**.

Unlike FID, which assumes Gaussian distributions of features, KID uses the **Maximum Mean Discrepancy (MMD)**, a **non-parametric kernel-based** measure, to directly compare the full distributions of real and generated features without parametric assumptions.

Like FID, both real and generated images are passed through a pretrained Inception v3 network to extract 2048-dimensional features. Then, KID computes the squared MMD between these sets of features:

$$\mathrm{KID}(X,Y) = \mathrm{MMD}^2(X,Y) = \mathbb{E}_{x,x'}[k(x,x')] + \mathbb{E}_{y,y'}[k(y,y')] - 2\mathbb{E}_{x,y}[k(x,y)],$$

where $k(x,y) = \left(\frac{1}{d}x^\top y + 1\right)^3$ is a polynomial kernel.

Interpretation: Lower FID = better: Generated distribution is closer to the real one.

Advantages over FID:

- **No Gaussian assumption** – fully non-parametric
- **Unbiased estimator**, even with small sample sizes

Results: Evaluation Metrics (IV)

CLIP Score evaluates how well a generated image aligns with its **textual description**. CLIP Score leverages the CLIP model, which jointly embeds images and text into a shared feature space.

Given a generated image x and its associated text t, CLIP encodes them into normalized embeddings e_x and e_t. The score is defined as their **cosine similarity**, scaled to the range $[0, 100]$:

$$\text{CLIPScore}(x, t) = \max\left(100 \cdot \frac{\langle e_x, e_t \rangle}{\|e_x\| \cdot \|e_t\|},\ 0\right)$$

Intuition:

- Measures **semantic consistency** between image and caption
- **Higher CLIP Score = better**: stronger image–text alignment

Strengths:

- Strong correlation with human judgments
- Works across domains without retraining

Limitations:

- Ignores image fidelity, realism, and diversity; therefore it should be combined with metrics like FID/KID for holistic evaluation

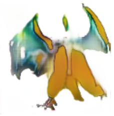
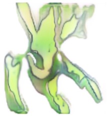
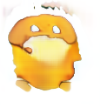
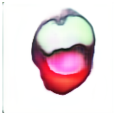
Results: Quantitative Results

Quantitative results on two held-out test sets:

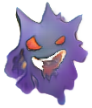
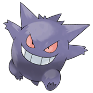
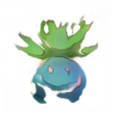
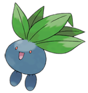
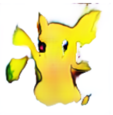
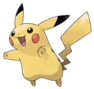
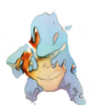
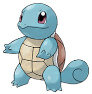
- **Intra-Test Set**: Designed to assess the model's robustness to lexical and syntactic variation. It contains test entries with paraphrased descriptions of training samples. These paraphrases were never seen during training and are used to evaluate the model's ability to generate accurate images when given semantically equivalent, but syntactically different text inputs.
- **Novel-Test Set**: Intended to measure the model's generalization capability. It includes entirely novel textual descriptions corresponding to unseen visual concepts – that is, none of these image-text pairs were present during training. This setup evaluates the model's capability to synthesize plausible outputs based solely on its learned textual and visual priors.

Metric	**Intra-Test**	**Novel-Test**
KID ($\downarrow$)	0.047 ± 0.00001	0.068 ± 0.00001
Inception Score ($\uparrow$)	2.936 ± 0.321	2.642 ± 0.244
FID ($\downarrow$)	107.404	117.837
CLIP Score ($\uparrow$)	25.416	24.580

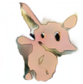
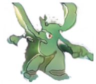
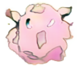
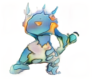
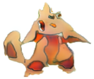
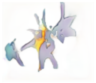
Results: Intra-Test Qualitative Evaluation (I)

Name	Original Description	Train Paraphrase	Test Paraphrase	Generated Image	Original Image
Charizard	A towering, dragon-like creature with an orange body, cream belly, and powerful wings with blue-green undersides. Its long neck and horned head add to its fearsome appearance. A blazing tail and sharp claws complete its intimidating, sky-dwelling form.	This creature is a formidable, dragon-like figure with imposing proportions. Its orange body, cream belly, and wingspan are impressively long and blue-green. Additionally, it has horned head and long neck, as well as blazing tail and claws that add to its intimidating appearance.	**It is a formidable, dragon-like creature with an orange body, curved shoulders, cream skin, and wings that have blue-green undersides. Its long neck and horned head are both impressive. A tail that is hot as lightning and its sharp claws complete its intimidating appearance.**		
Scyther	A large, green insectoid with scythe-shaped forearms, large wings, and sharp mandibles. Its sleek body and agile form enable swift, slicing movements.	This insectoid is a large green insect with broad wings, scythe-shaped forearms, and sharp mandibles. Its agile posture and smooth body make it easy to slice through its body quickly.	**A slender green insectoid with broad wings, forearms that resemble a hawk, and sharp mandibles. Its agile posture and smooth muscles allow for quick, blade-crushing actions.**		
Psyduck	A duck-like creature standing on two legs, with yellow fur, a round body, and a wide cream beak. Its blank stare, tiny pupils, and three black hairs atop its head give it a perpetually dazed and confused look.	This is a duck-like creature standing on two legs, with yellow fur covering its entire body and curved cream beak. Its uncomplicated expression, minute pupils, and three black hairs on its head make it appear constantly disoriented and dazed.	**A duck-like creature with yellow fur, a round body and wide cream beak. Its eyes are blank, its pupils are small, and its head is covered in black hair. The creature stands on two legs and has an uncontrollable countenance.**		
Electrode	A smooth, spherical form resembling an inverted Poké Ball with a red bottom half and white top. It has a wide mischievous grin and a glossy, polished surface that gleams brightly.	A slick, round shape resembling an inverted Poké Ball with a red bottom half and white top. It has pronounced cheekbones and gleaming eyes and shiny, polished surfaces.	**It is a smooth, round shape that looks like an inverted Poké Ball with 'the bottom half red and the top white' A wide grin: All over shiny and polished.**		

Results: Intra-Test Qualitative Evaluation (II)

Name	Original Description	Train Paraphrase	Test Paraphrase	Generated Image	Original Image
Gengar	A round, purple figure with a wide, wicked grin and sharp pointed ears. Its bright red eyes gleam with mischief, and spiky protrusions line its back, enhancing its eerie, playful aura.	A purple-colored object with a wide, wicked smile and pointed, sinister ears. Its brilliant red eyes are filled with mischief, and its back is lined with sharp, jagged protrusions.	**An enigmatic, purple creature with a wide, wicked grin and pointed ears. Its bright red eyes are filled with mischief, and its back is lined with sharp indentations of spiky hair.**		
Oddish	A tiny, round creature with deep blue skin, glowing red eyes, and a small smiling mouth. From the top of its head sprout five wide, leafy green blades, giving it the appearance of a walking plant bulb or animated weed emerging from the earth.	A small, round animal with deep blue skin, luminous red eyes, and a tiny smile. Its head is covered in five broad green blades, giving it the appearance of shedding weed or bulbous plant bulbs.	**This is a small, circular animal with deep blue skin, red gleaming eyes and minuscule smile. Five broad, leafy green blades grow from the tip of its head, giving it an appearance of pulverized weed or ambulatory plant bulbs.**		
Pikachu	A small, round-cheeked rodent with bright yellow fur and two brown stripes on its back. Its long ears have black tips, and its face features large eyes and red circular cheeks. A jagged, lightning-shaped tail with a brown base gives it a lively, energetic silhouette.	This is a small, round-cheeked, bright yellowish-fur rodent with two brown stripes on its back with long, black tips of fur. Its face has large eyes and red circular cheeks and twirling eyes, while its jagged, lightning-shaped tail with striped legs creates an energetic, look that can be worn all day.	**With a bright yellow fur and two brown stripes on its back, this small rodent has long ears with black tips. Its face is distinguished by large eyes and red circles on the cheeks. A jagged, lightning-shaped tail with contrasting brown base gives it vigor.**		
Squirtle	A small turtle-like creature with light-blue skin, big purplish eyes, and a curled tail. Its sturdy brown shell has a pale yellow underside and thick white edging, giving it a tough, playful look.	A tiny turtle-like creature with a light-blue complexion, sizable purplish eyes, and curved tail. Its sturdy brown shell has accumulated globules on its underside, which are now covered in contrasting white paint, creating an aggressive and playful appearance.	**The small, turtle-like creature has large purplish eyes and a curled tail. Its sturdy brown shell has pheromone patches on its underside and thick white bordering, giving it ferocity.**		

Results: Novel-Test Qualitative Evaluation

Name	Description	Generated Image	Original Image
Buneary	Small and rabbit-like with curled brown ears. Its upper body has smooth chocolate-colored fur, while the lower half is covered in light tan fleece.		
Cacturne	A humanoid cactus resembling a scarecrow with spiky limbs, green rhombi patterns, a grinning face, and a spiked triangular hat-like head.		
Cleffa	A small, pink creature with a star-like shape and curled tail. It has stubby limbs, a cheerful face, and a swirl on its forehead, giving it a charming and innocent appearance.		
Croconaw	A stout, blue reptilian creature with a yellow jaw and red spikes on its head, back, and tail. Its body has an uneven blue-and-yellow pattern, and its large jaws are filled with slanted fangs.		
Darmanitan	A squat, ape-like figure with a red body and tan face. It has bushy flame-shaped eyebrows, spiked teeth, and strong limbs for an energetic appearance.		
Kyurem	A towering gray dragon with icy features, jagged wings, yellow glowing eyes, and shards of ice along its body.		

Conclusions & Future Work (I)

This project presented a text-to-image synthesis model tailored to generate Pokémon-style sprites from Pokédex descriptions.

Key contributions:

- A custom dataset addressing limitations of the original corpus.
- A conditional GAN architecture combining global and token-level textual conditioning (via a fine-tuned text encoder).

The model captures coarse visual attributes (e.g., color, shape), but struggles with fine-grained details and spatial accuracy.

Improvements:

- **Model improvements:**
  - Actually token-level conditioning is applied only in the generator to limit computational overhead; extending it to the discriminator is a promising next step.
  - Integration of *self-attention* mechanisms within both generator and discriminator could improve spatial coherence and fine detail generation. Such mechanisms were not included in the present work due to their well-known quadratic computational complexity, which poses practical challenges given limited resources.

Conclusions & Future Work (II)

- **Data-related improvements:**
  - Expand dataset size to improve generalization and diversity.
  - Simplify and standardize descriptions to reduce ambiguity.
  - Explore adaptive data augmentation methods (e.g., DiffAugment, ADA) to stabilize training in low-data settings.
- **Loss functions:**
  - Incorporate CLIP loss for better semantic alignment loss and perceptual loss for better visual fidelity.
- **Optimization:**
  - Perform Bayesian hyperparameter search for efficient training configuration discovery.