Politecnico di Bari

Deep Learning - Instructor: Vito Walter Anelli, Ph.D.
M.D. in Computer Engineering - Politecnico di Bari
Test code: 2025_VI - Released on July 2nd 2025
Deadline: August 3rd, 2025

I PikaPikaGen: Generative Synthesis of Pokémon Sprites from Textual Descriptions

1 Task Overview

Text-to-Image synthesis is a challenging and influential task in computer vision and natural language processing. The goal is to develop a model that can generate a stylized image from a given textual description. This task requires a deep semantic understanding of both the text and the visual domains, and the ability to map abstract textual concepts to concrete pixel representations.

In this project, we will tackle a specific instance of this task: generating Pokémon sprites based on their Pokédex descriptions. The model will take a textual description detailing a Pokémon's appearance, type, and characteristics, and synthesize a corresponding 2D sprite that accurately reflects these attributes. This involves learning the intricate relationship between the vocabulary used in the descriptions and the visual features of the Pokémon, such as shape, color, and texture.

IMPORTANT: Create a live, interactive demo of the trained model using Gradio. The demo should provide a simple web interface where a user can input a textual description and see the generated Pokémon sprite.

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

Figure 1: Conceptual overview of the Text-to-Pokémon generation task. (AI-generated image using `Imagen-4-Ultra`)

2 Objective

The primary objective of this project is to design, implement, and train a generative model capable of creating Pokémon sprites from textual descriptions. The core architecture will be an Encoder-Decoder model with an Attention mechanism.

Sub-Objectives

The specific goals are to:

- **Implement a Transformer-based Encoder** to process the input text. The encoder's embedding layer will be **initialized with pre-trained bert-mini vectors** (256 dimensions) and will be fine-tuned during training to adapt to the Pokémon-specific domain.
- **Develop a Convolutional Neural Network (CNN) based Decoder (Generator)** that synthesizes the sprite image from a latent representation.
- **Integrate an Attention Mechanism** that allows the decoder to selectively focus on the most relevant words from the input description while generating different parts of the image.
- Train the model end-to-end on a dataset of Pokémon sprites and their corresponding descriptions, and evaluate its performance both qualitatively and quantitatively.

3 Dataset

For this assignment, the students will use the "The Pokémon Dataset", which is publicly available and well-structured for this task. This dataset contains comprehensive information for each Pokémon, including sprites and textual descriptions.

Dataset Link: `https://github.com/cristobalmitchell/pokedex/`

This repository containins:

- **Textual Descriptions:** Detailed Pokédex entries describing each Pokémon's appearance, abilities, and nature.
- **Sprite Images:** High-quality, 215×215 pixel sprite images with transparent backgrounds, providing a consistent visual target for the model.
- **Metadata:** Additional information such as type, name, and stats, which can be used for more advanced conditional generation.

4 Data Preprocessing

Before training, the data must be preprocessed and converted into a suitable format for the deep learning model.

Text Preprocessing:

- Parse the CSV file to extract the textual descriptions for each Pokémon.
- Clean and tokenize the text using **bert-mini**. Use its vocabulary of all unique tokens and map each token to its integer index. Optionally pad all text sequences to a fixed length to ensure uniform input size for the encoder.

Image Preprocessing:

- Load the 215×215 sprite images.
- **[Optional, useful if the students meet problems]** Normalize the pixel values to a specific range, such as [-1, 1], which is often beneficial for training generative models.
- Handle the alpha (transparency) channel appropriately. A common approach is to convert the image to RGB by blending it with a white background.

5 Model Architecture

High-level sketch of the suggested Encoder-Decoder with Attention architecture.

Text Encoder:

- **Embedding Layer:** This layer maps the input token indices to dense vectors. It should be initialized with pre-trained bert-mini (256-dim) embeddings and fine-tuned during training.
- **Transformer Encoder:** The sequence of embeddings is fed into a small Transformer encoder (consisting of multi-head self-attention and feed-forward layers). The encoder's role is to produce a sequence of contextual hidden states, one for each input token.

Attention Mechanism:

- The attention mechanism links the decoder's state with the encoder's output. At each step of the image generation process, it calculates a context vector by computing a weighted average of the encoder's hidden states. This allows the decoder to ”pay attention” to specific words in the description (e.g., ”fiery tail,” ”blue wings”) when generating the relevant parts of the sprite.

Image Decoder (CNN Generator):

- The decoder is a Convolutional Neural Network that generates the image. It should take an initial random noise vector concatenated with the text-derived context vector as input.
- It will use a series of Transposed Convolution layers to upsample the spatial dimensions from the initial latent vector to the final 215x215 image size.
- Each transposed convolution layer should be followed by a Normalization Layer (to stabilize training) and an activation function.

6 Hints

You can:

- organize the dataset to fit the training needs;
- if necessary, use techniques to augment the dataset;
- The choice of loss function is critical for generative models. A good starting point is a pixel-wise Reconstruction Loss, such as L1 (Mean Absolute Error) or L2 (Mean Squared Error), between the generated image and the ground-truth image. L1 loss is often preferred as it tends to produce less blurry images than L2.

7 Training and Evaluation

Train the model on the training set using suitable loss functions and optimization techniques. Monitor the training process by evaluating the model's performance on the validation set. Use appropriate evaluation metrics. Fine-tune the model hyperparameters to improve its performance on the validation set. Finally, evaluate the trained model on the test set and report its performance metrics.

8 Model Analysis

Perform an analysis of the model's performance. Identify any challenges or limitations faced by the model. Provide recommendations for further improvements or modifications to enhance the model's accuracy and efficiency.

9 Deliverables

- A well-commented Python code implementation of the models.
- A report documenting the detailed approach taken to solve the problem, including data preprocessing, model architecture, training, and evaluation results.
- Analysis and discussion of the model's performance, challenges faced, and recommendations for improvement.
- A presentation that will be discussed at the time of the Oral Exam.

Notes

You are encouraged to utilize existing deep learning libraries such as TensorFlow or PyTorch for implementing the model. Make sure to properly document your code and provide clear explanations in the report to demonstrate your understanding of the concepts and techniques used.