Text-to-image model for Pokemon images

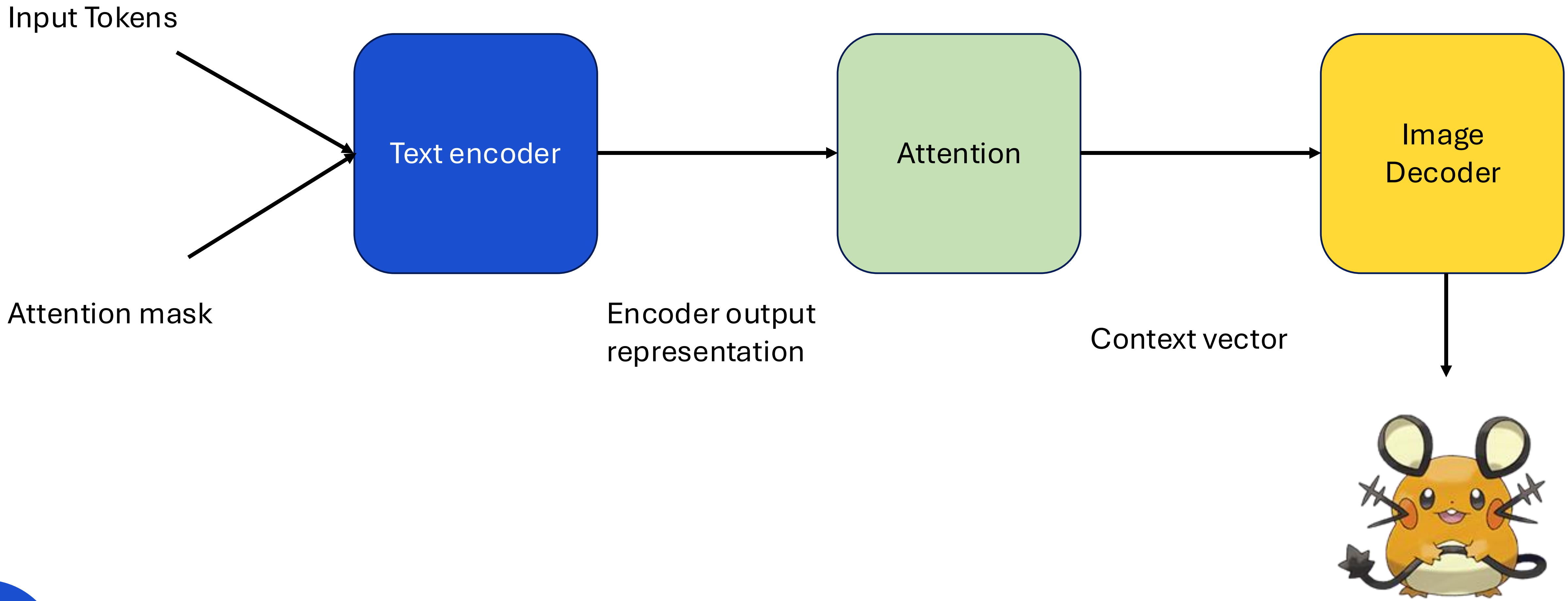
Mattia Tritto

Master Degree in Computer Engineering

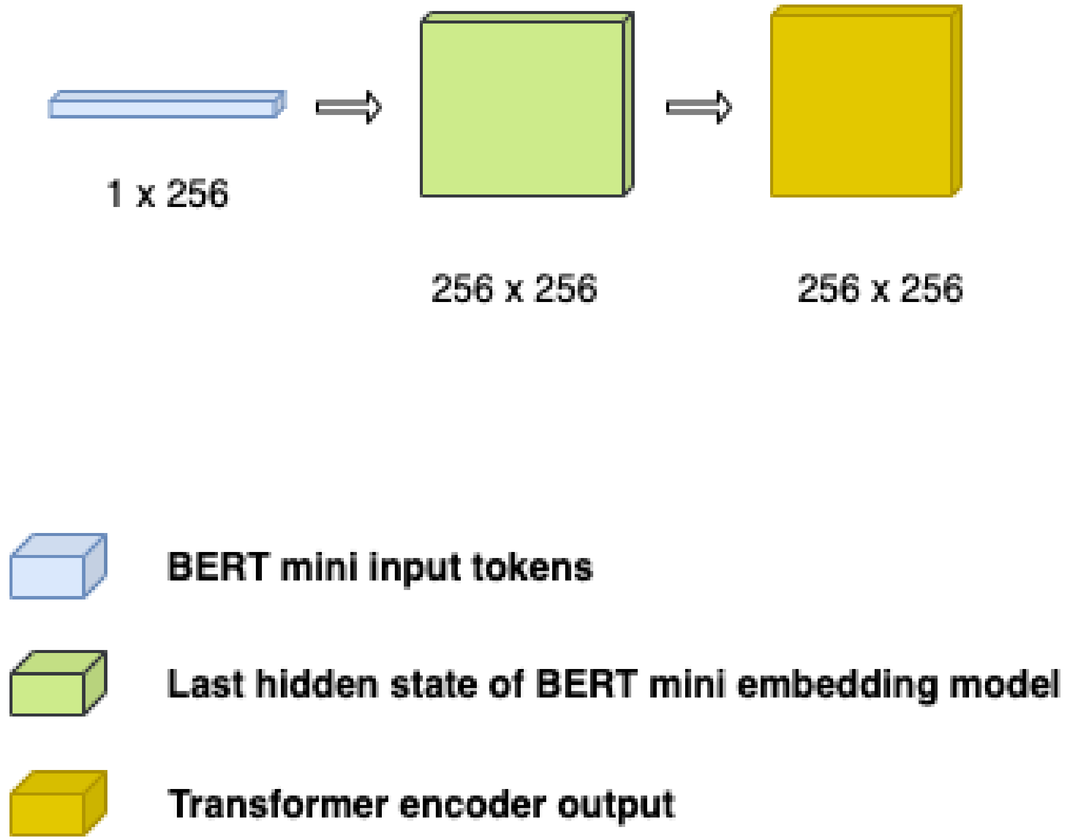
Model architecture

A visual explanation of the model architecture

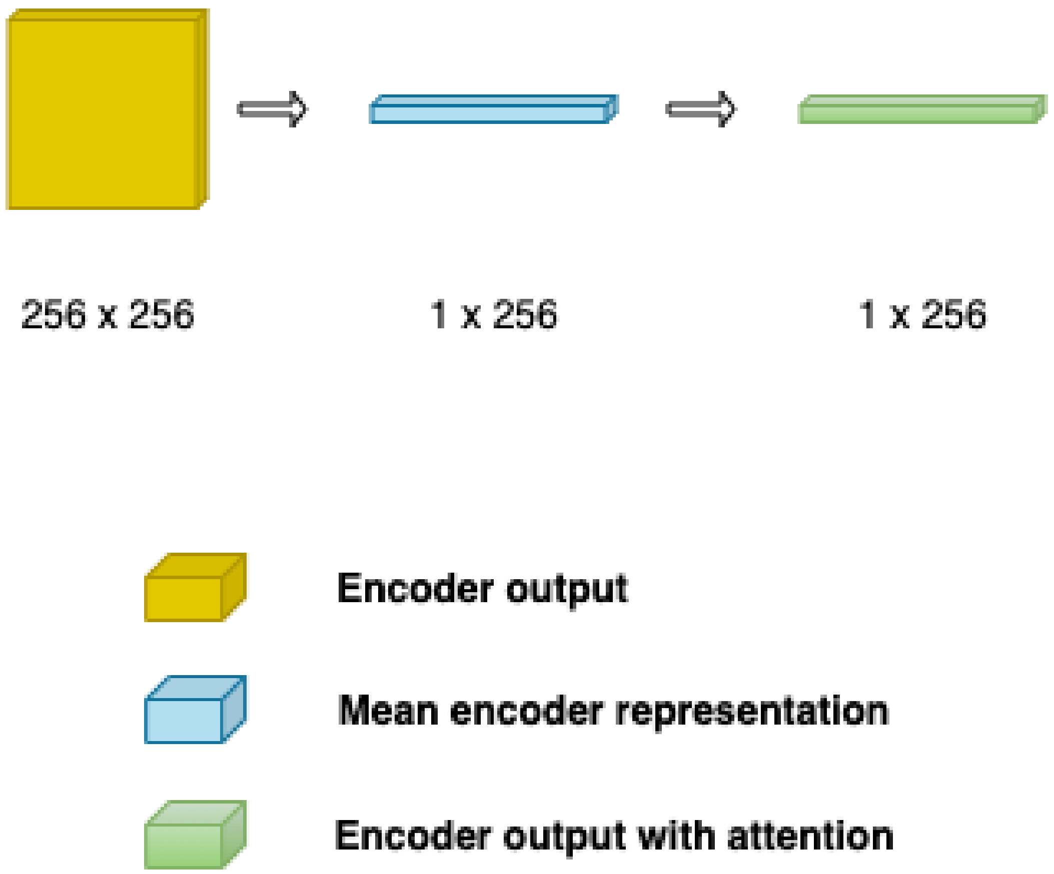
A high-level overview



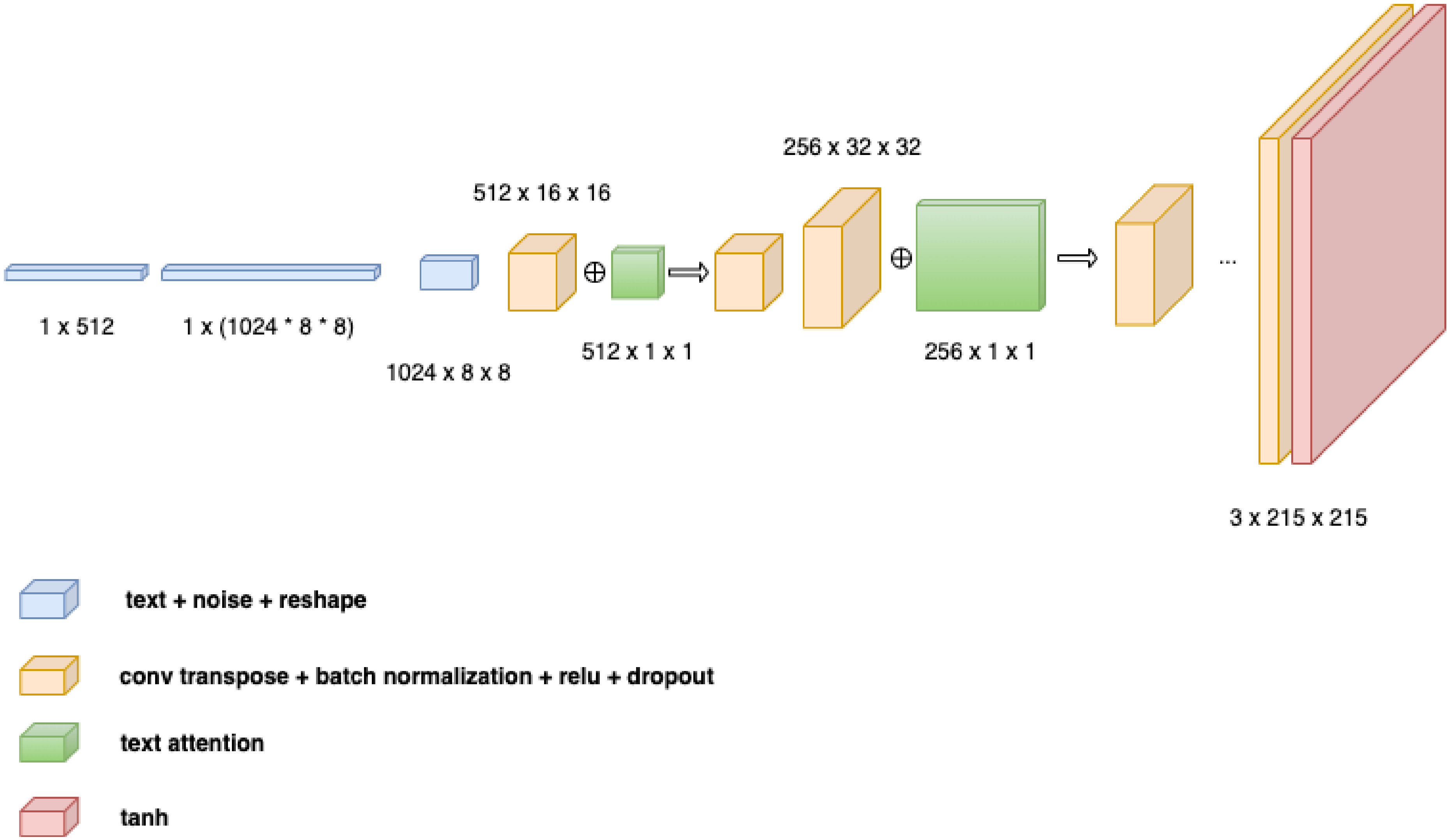
Text encoder architecture



Attention module



Decoder module



Preprocessing

Data preparation for the model

Images preprocessing

- For preprocessing, I first extracted **the alpha channel** to identify transparent regions in the image. These regions are then **filled with white** to convert the image into a standard RGB format.
- Next, I **apply min-max normalization** to scale the pixel values from $[0,255]$ to $[-1,1]$, which aligns with the output range of the **decoder's final Tanh activation function**.

Image augmentation

- To increase the size of the dataset, I **applied image augmentation techniques,** specifically, by **rotating the images**. This should help **improve model generalization** by introducing variability in the training data.



Text preprocessing

- For text preprocessing, I create an enriched textual description by combining multiple fields from the dataset: **description**, **primary type**, **secondary type**, and **classification**. These additional fields are included because they often carry **visual attributes** that can enhance the model's ability to generate images aligned with the textual input.

- *«There is a plant seed on its back right from the day this Pokèmon is born. The seed slowly grows larger. This Pokémon is classified as a Seed Pokèmon. It is of grass type and poison type»*

Training process

Some details about the training

L1 Loss

- As the primary loss function for evaluating the model, L1 loss was used. This **pixel-wise loss measures the absolute difference between the predicted and ground truth images**. L1 loss is particularly effective in preserving fine **accurate reconstruction at each pixel details and sharpness in generated images**, making it suitable for tasks that require high-fidelity outputs.

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = \left| x_n - y_n \right|,$$

CLIP score

The CLIP score is a metric that evaluates how well a generated image aligns with a given text description. It is based on the **CLIP model (Contrastive Language-Image Pretraining)** developed by OpenAI, which **learns joint embeddings for images and text in a shared semantic space**.

To compute the CLIP score:

- The generated image and its corresponding text description are both passed through the pretrained CLIP model.
- The model extracts feature embeddings for both modalities.
- The **cosine similarity between these embeddings** is calculated.

Results

All the experiments conducted

First trial configuration

Data augmentation used
False

Epochs
100

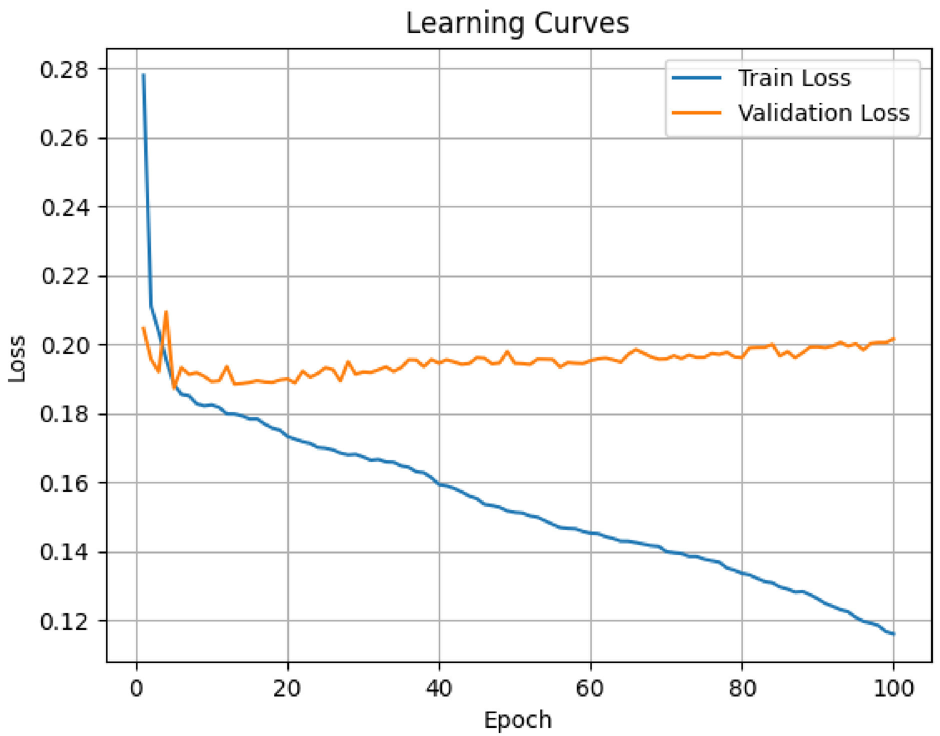
Loss
L1

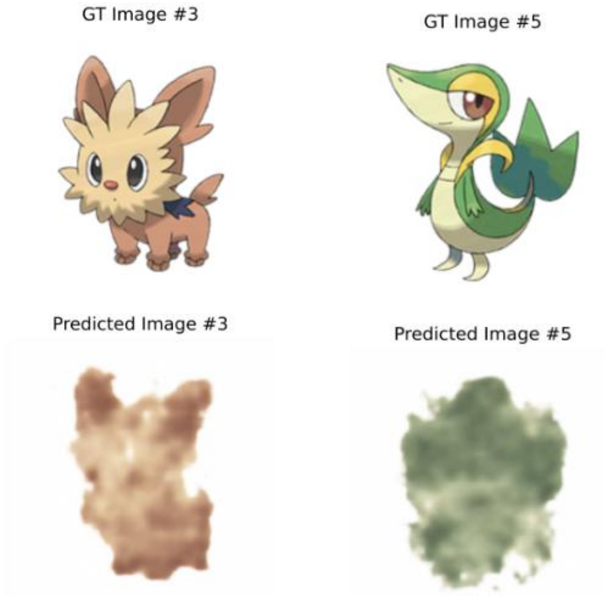
Bath size
10

Dropout
False

Learning rate
1e-4

First trial





(a) Example on training set (ID 1)

(b) Example on validation set (ID 1)

Figure 3.2: Generated Pokémons from experiment ID 1 on training and validation sets.

First trial

- As observed from the learning curves, the model is able to learn effectively during training; however, it quickly overfits the data. This is evident as the **training loss decreases steadily**, while the **validation and test losses remain higher**, indicating **poor generalization to unseen data**.

- Qualitatively, the model performs well on the training images, successfully learning their specific visual patterns. However, it fails to replicate similar quality on the validation images, suggesting that it **has memorized the training set rather than learning generalizable features**.

Second trial configuration (dropout added)

Data augmentation used
False

Epochs
100

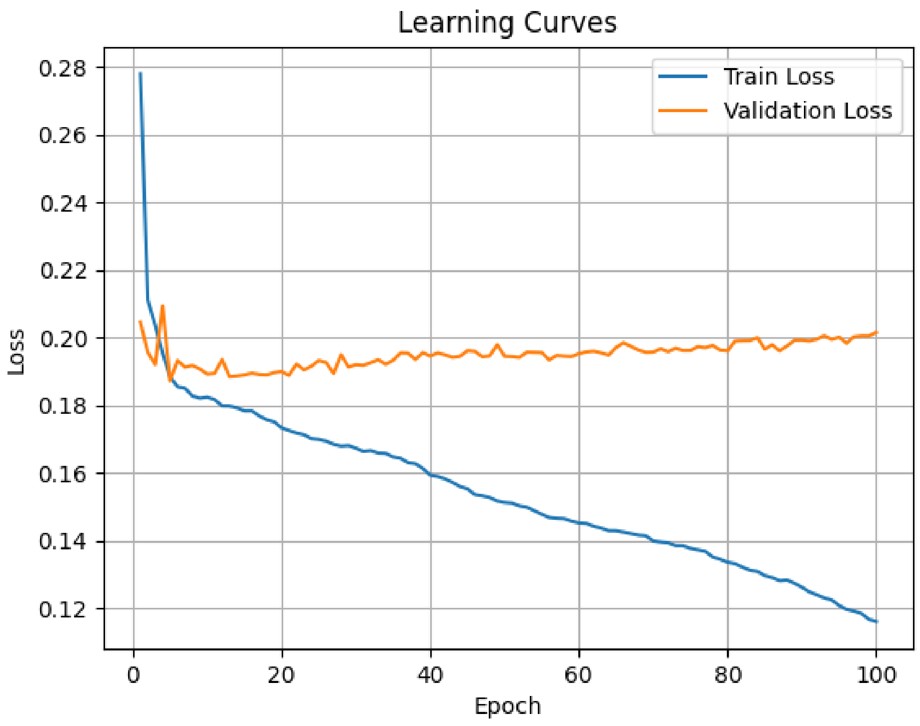
Loss
L1

Bath size
10

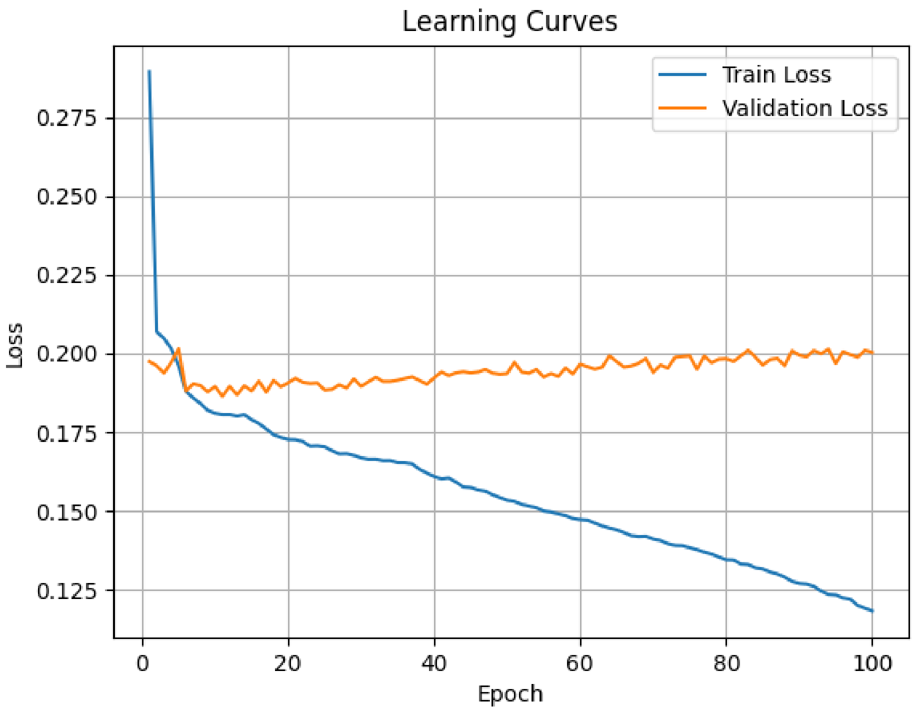
Dropout
0.3

Learning rate
1e-4

Comparison between learning curves



First trial learning curve



Second trial learning curve

Second trial (dropout added)

- According to the results**, dropout did not improve the model's ability to generalize to unseen images on the validation and test sets**. One possible explanation is that the model may already have sufficient regularization or capacity, and introducing dropout might not have helped prevent overfitting.

Third trial configuration (CLIP loss)

Data augmentation used
False

Epochs
100

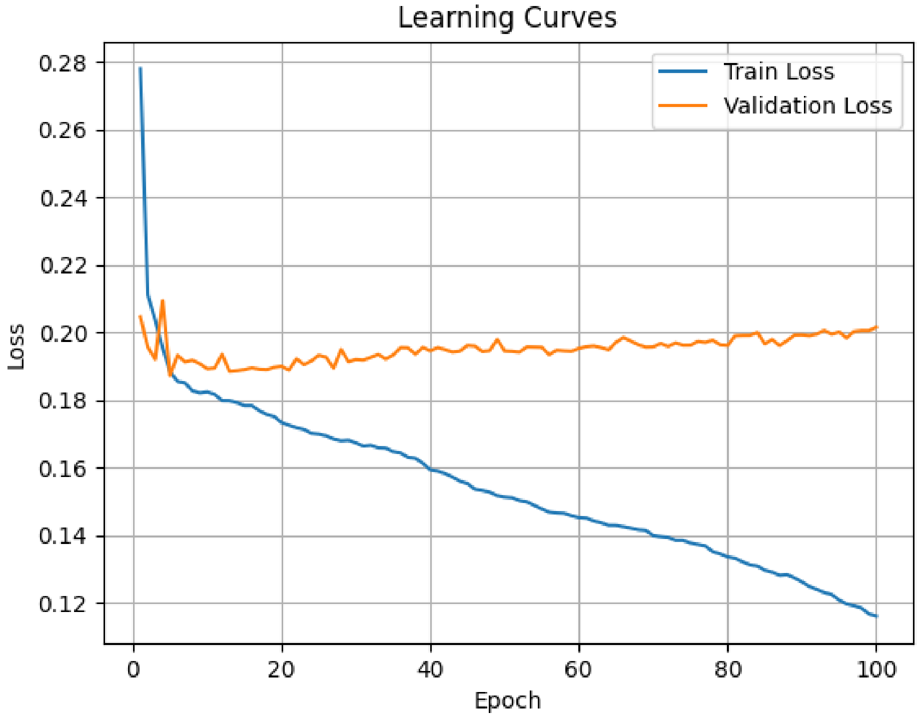
Loss
CLIP loss

Bath size
10

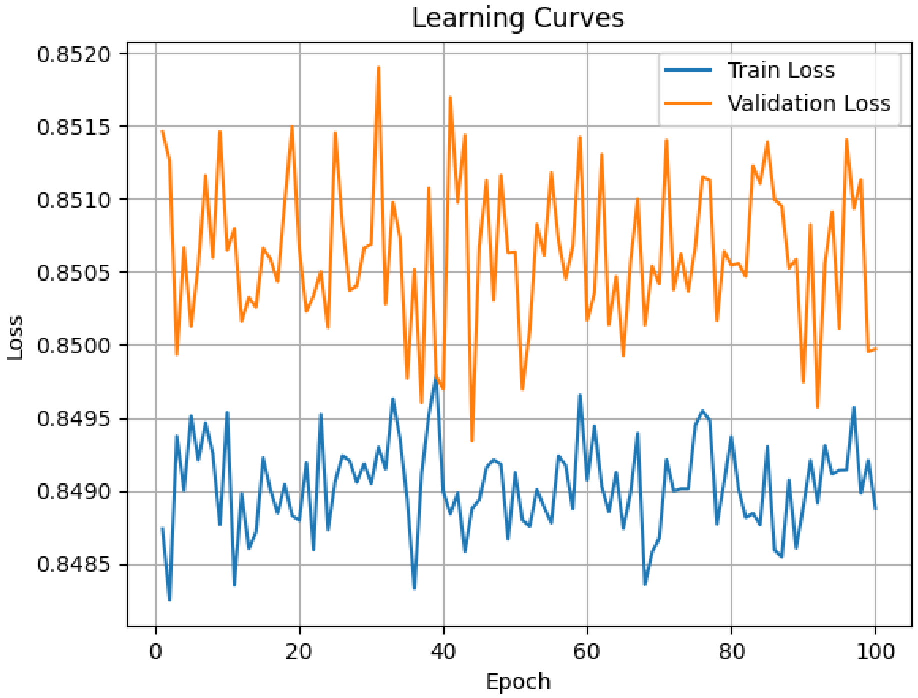
Dropout
False

Learning rate
1e-4

Comparison between learning curves



First trial learning curve



Third trial learning curve

Third trial (CLIP loss)

- According to the results, the **CLIP loss leads to unstable training**, as the model fails to learn, evidenced by the fact that **the training loss does not decrease**. One possible reason for this behavior is the initialization strategy: instead of using BERT-mini vectors, the model should be initialized with embeddings from the CLIP model itself. This would ensure **better compatibility with the CLIP loss** and facilitate more effective learning throughout the pipeline.

Fourth trial configuration (Data augmentation)

Data augmentation used
True

Epochs
100

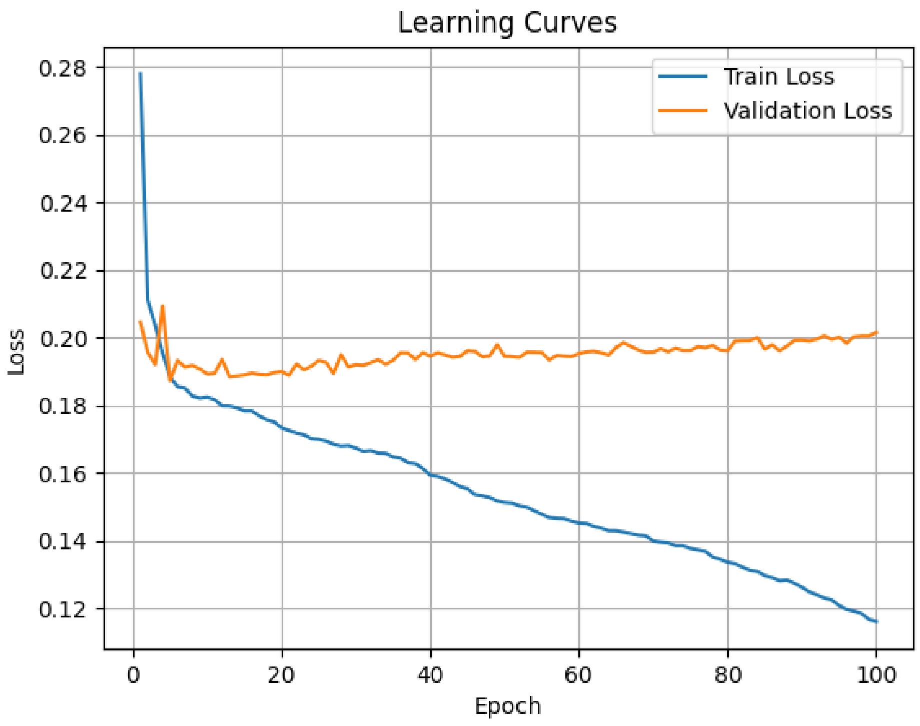
Loss
L1 loss

Bath size
10

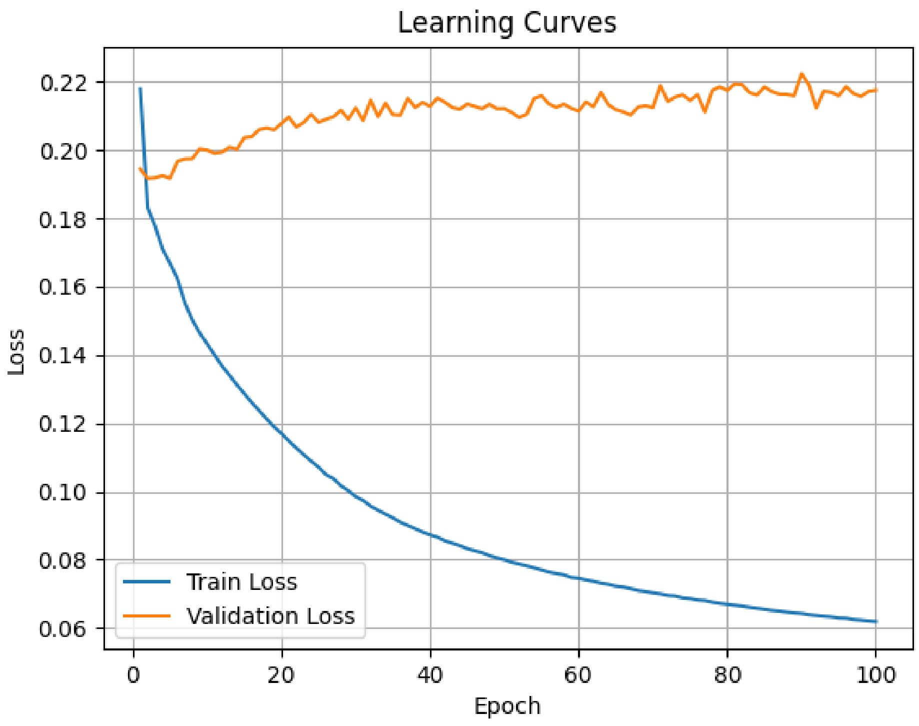
Dropout
False

Learning rate
1e-4

Comparison between learning curves



First trial learning curve



Fourth trial learning curve

Fourth trial configuration (data augmentation)

- According to the results, **augmenting the dataset did not help the model generalize better**. The performance on the validation set, both visually and based on evaluation metrics, remained almost the **same as without augmentation**. Moreover, the learning curves suggest that the model may have **overfitted the training data**, as the training loss decreased significantly while the validation performance did not improve.

Fifth trial configuration (enriched description)

Data augmentation used
False

Epochs
100

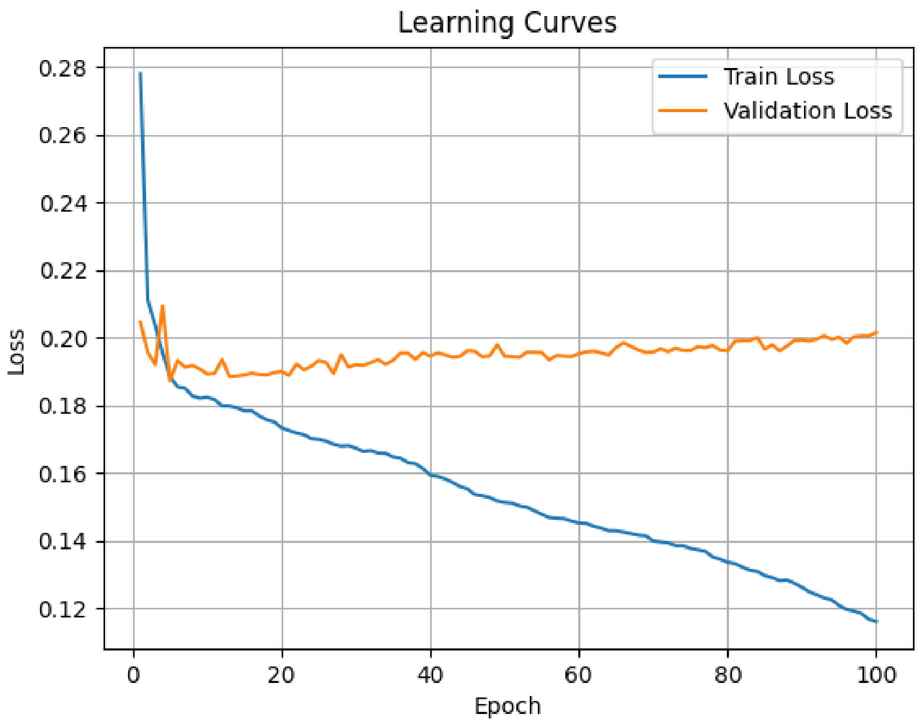
Loss
L1 loss

Bath size
10

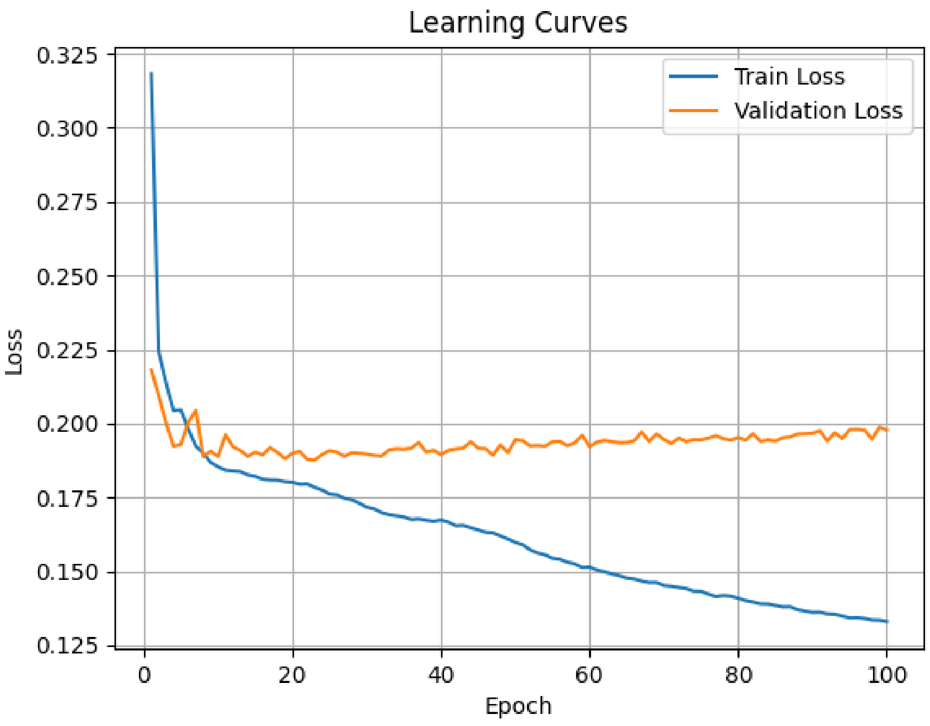
Dropout
False

Learning rate
1e-4

Comparison between learning curves



First trial learning curve



Fifth trial learning curve

Fifth trial configuration (enriched description)

- According to the results, using an **enriched description did not improve the model’s ability to generalize**, as the validation loss remained unchanged.

Hyperparameter optimization (1/4)

- As a final experiment, I conducted a **grid-search-style exploration of several hyperparameters** to identify the best configuration. The hyperparameters evaluated include:
  - **Learning rate** (1e-4, 1e-5);
  - **Weight Decay** (1e-5, 1e-6);
  - **Number of attention head in the encoder** (2, 4);
  - **Dimension of the feedforward encoder** (512, 1024);
  - **Number of transformer encoder layers** (2, 3).

Hyperparameter optimization (2/4)

Learning rate	Weight Deacy	N. heads attention encoder	Dim. feedforward encoder	Transfomer encoder layers	Best val. loss L1	Test loss L1	CLIP score
0,00010	0,000001	2	512	2	0,188	0,204	0,229
0,00010	0,000001	2	512	3	0,185	0,205	0,230
0,00010	0,000001	2	1.024	2	0,185	0,204	0,232
0,00010	0,000001	2	1.024	3	0,187	0,205	0,232
0,00010	0,000001	4	512	2	0,186	0,208	0,228
0,00010	0,000001	4	512	3	0,187	0,209	0,232
0,00010	0,000001	4	1.024	2	0,188	0,204	0,232
0,00010	0,000001	4	1.024	3	0,186	0,203	0,226
0,00010	0,000010	2	512	2	0,187	0,208	0,227
0,00010	0,000010	2	512	3	0,187	0,207	0,227
0,00010	0,000010	2	1.024	2	0,187	0,205	0,230
0,00010	0,000010	2	1.024	3	0,185	0,205	0,228
0,00010	0,000010	4	512	2	0,187	0,205	0,233
0,00010	0,000010	4	512	3	0,185	0,204	0,229
0,00010	0,000010	4	1.024	2	0,185	0,205	0,226
0,00010	0,000010	4	1.024	3	0,187	0,205	0,227
0,00001	0,000001	2	512	2	0,192	0,211	0,224
0,00001	0,000001	2	512	3	0,193	0,210	0,228
0,00001	0,000001	2	1.024	2	0,192	0,212	0,223
0,00001	0,000001	2	1.024	3	0,191	0,215	0,226
0,00001	0,000001	4	512	2	0,191	0,211	0,225
0,00001	0,000001	4	512	3	0,193	0,212	0,227
0,00001	0,000001	4	1.024	2	0,198	0,216	0,228
0,00001	0,000001	4	1.024	3	0,198	0,215	0,229
0,00001	0,000010	2	512	2	0,190	0,212	0,223
0,00001	0,000010	2	512	3	0,196	0,219	0,228
0,00001	0,000010	2	1.024	2	0,195	0,208	0,229
0,00001	0,000010	2	1.024	3	0,191	0,217	0,227
0,00001	0,000010	4	512	2	0,193	0,216	0,228
0,00001	0,000010	4	512	3	0,193	0,213	0,230
0,00001	0,000010	4	1.024	2	0,191	0,215	0,227
0,00001	0,000010	4	1.024	3	0,191	0,209	0,226

Hyperparameter optimization (3/4)

- As shown in the previous table, the performance across different configurations **remains relatively consistent**. Specifically, the validation loss ranges between 0.18-0.19, the test loss between 0.20-0.21, and the CLIP score between 0.20-0.22. This suggests that the overall performance **is not highly sensitive to hyperparameter variations** within the explored ranges.

Hyperparameter optimization (4/4)

- Nevertheless, if a single configuration must be selected, I recommend the one that achieved the **lowest validation and test loss:**

- Learning Rate: 1×10^{-4}
- Weight Decay: 1×10^{-5}
- Loss Function: L1 Loss
- Epochs: 100
- Batch Size: 10
- Augmentation: Enabled (N=8)
- Enriched Description: Yes
- Transformer Encoder Layers: 2
- Attention Heads: 4
- Feedforward Dim: 1024
- Dropout encoder: 0.3
- Dropout attention: 0.3
- Dropout decoder: 0.3

Conclusions

Following a series of trials, the key takeaways are...

Conclusions (1/4)

- In conclusion, the current architecture **does not appear to be particularly effective for generating Pokémon sprites**. Across all configurations, the model consistently **exhibits a tendency to overfit**, failing to generalize well on both the validation and test sets: this represents a significant limitation.

- I strongly believe that **the core issue lies in the architecture itself**. To achieve meaningful improvements, it would be necessary to redesign it entirely. The most promising direction would be to adopt a **Stable Diffusion-based model**, which has been extensively studied and proven effective for generating images from textual prompts.

Conclusions (2/4)

(a) Prediction image (b) Ground truth image

Figure 3.7: Training set example, prediction VS ground truth

(a) Prediction image (b) Ground truth image

Figure 3.8: Another training set example, prediction VS ground truth

Conclusions (3/4)

(a) Prediction image (b) Ground truth image

Figure 3.9: Test set example, prediction VS ground truth

(a) Prediction image (b) Ground truth image

Figure 3.10: Another test set example, prediction VS ground truth

Conclusions (4/4)

- However, before proceeding with a complete architectural overhaul, there are still **a few avenues worth exploring** to attempt incremental improvements. I report here 3 ones:

- Paraphrasing textual descriptions
- A combined loss function that involves L1 and CLIP
- Staged training

Thank You