Pokémon Variational Autoencoder for Latent Diffusion

Braedan Kennedy: bkenne07@calpoly.edu

Briana Kuo: brkuo@calpoly.edu

Luis David Garcia: lgarc120@calpoly.edu

Paul Jarski: pjarski@calpoly.edu

CSC 587-01 Professor Ventura

Table of Contents

Introduction

Our goal was to use available Pokémon sprite data to create a latent diffusion model for the purpose of generating original sprites that could be used to extend open-source software such as Pokémon Showdown! [6]. This topic would allow us to apply many of the concepts covered in CSC 587, but in a more challenging context. Unlike the Frey dataset [1], which contains a standardized set of one human with different facial expressions, these Pokémon sprites represent a great deal of variation. Although we ended up with a large dataset of around 26,000 sprites, we may still not have had enough data for the model to learn to spontaneously generate quasi-human renderings of novel Pokémon. Nevertheless, we made many discoveries about the strengths and weaknesses of various models, and the course of our experiments demonstrates progress in our thinking about image generation problems.

Figure 1. Our Variational Autoencoder's Reconstructions of 128x128 Pokémon Images.

Our final model was a Variational Autoencoder for Latent Diffusion. This work is non-trivial in the sense that generating high quality images from a relatively sparse and heterogeneous dataset is a challenging task. In this work, our team was able to perform preprocessing to obtain a useable dataset, create a Variational Autoencoder capable of encoding and decoding Pokémon character sprites with a minimal loss of information, and train a Latent Diffusion model to generate new character sprites inspired by the style of official Pokémon. The Latent Diffusion based on the Variational Autoencoder showed great improvement over the first version, based on a regular Autoencoder.

Related Work

L. Horsley and D. Perez-Liebana developed a sprite generator using a Generative Adversarial Network (GAN), with both the generator and discriminator consisting of Multilayer Perceptron (MLP) networks [4]. They identified the main limitation as the requirement for diverse datasets, noting their model's difficulty in generating high-quality results due to the bold and diverse characteristics needed, indicating a necessity for model enhancement.

J. J. Chu, Z. Shan, and X. Sun designed a convolutional network for 64x64 RGB Pokémon images, utilizing ReLU and SoftMax activations to classify Pokémon into their respective types (e.g., fire, water, fighting) [2]. However, their model underperformed on the test set. Despite this, they offered valuable insights into using data augmentation, such as image rotation, to improve the model's generalizability.

Manakov, Rohm, and Tresp also experimented with Convolutional Autoencoders and Pokémon sprites, using a much smaller dataset, and warned against decreasing the bottleneck size too much as this causes overfitting and poor validation performance [7]. We balanced this concern against the difficulties of training a diffusion model to learn the latent distribution of our model. Eventually we resorted to using a Variational Autoencoder to make the latent distribution easier for the diffusion model to learn, and to encourage similar sprites' latent vectors to cluster.

Perez et al. introduced their diffusion variational autoencoder, aimed at capturing the topological and geometrical properties of datasets via a novel approach that permits an arbitrary manifold as the latent space [9]. While their methodology holds promise for data structure understanding, their primary emphasis on the simpler MNIST dataset, which is far less complex than Pokémon imagery, highlights the need for models adept at managing the heightened complexity and diversity of Pokémon datasets. At 60,000 images, the MNIST dataset is also much larger than the 26,000 images we used for our experiments.

Rombach et al. presented a seminal paper on latent diffusion for image synthesis, proposing a foundational architecture that generates images from a UNET embedding within a latent diffusion model. Notably, their approach to KL-regularization influenced our use of a Multilayer Perceptron (MLP) model [11]. However, to achieve computational efficiency, we chose a Variational Autoencoder over their more resource intensive UNET architecture, demonstrating a strategic adaptation to balance performance and computational demand. We also knew what to expect from a Variational Autoencoder, whereas a UNET would have introduced many additional considerations, like the number and size of the skip connections. The few experiments we performed with a UNET were not promising.

Methodology

Dataset and Preprocessing

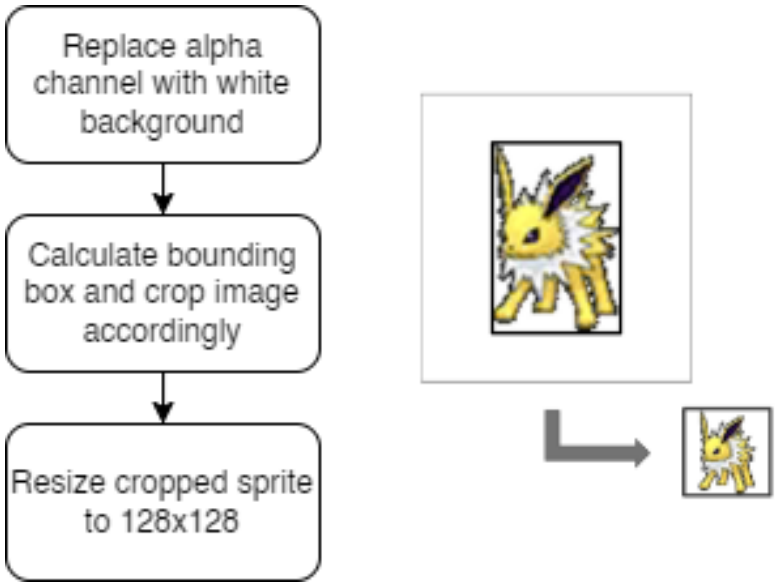

Figure 2. Dataset Preprocessing Methodology.

The source of our Pokémon sprites is from a project called PokeAPI [10], which contains RGBA images of every Pokémon from every Pokémon game. Since the style of the character art varies between different versions of the game, we selected a subset of version of the game (4 through 7), which all utilize a similar sprite style. We had issues with corrupted images files, so we had to implement a blacklist filter to avoid loading those images. After applying the blacklist, this resulted in approximately 13 thousand unique sprites. In addition to these ~13,000 unique sprites, we performed data augmentation on these images to increase the amount of data. We simply flipped the sprite along the vertical axis, effectively doubling the size of our dataset while still retaining high quality images.

Analysis of the sprites from these generations revealed a range of sprite sizes from 80x80 to 256x256 (px), and a subsequent need to standardize them. The solution implemented in this work was to first remove the alpha channel from a given image, then replace it with a white background. The next step was to implement a minimal bounding box detector and use it to locate the sprite within the larger image. Once the bounding box was obtained, the final step was determining the smallest square containing the bounding box by creating a square based on the largest extent of the bounding box, cropping out of the original image, and resize it to the desired dimensions – in our case, 128x128.

Variational Autoencoder

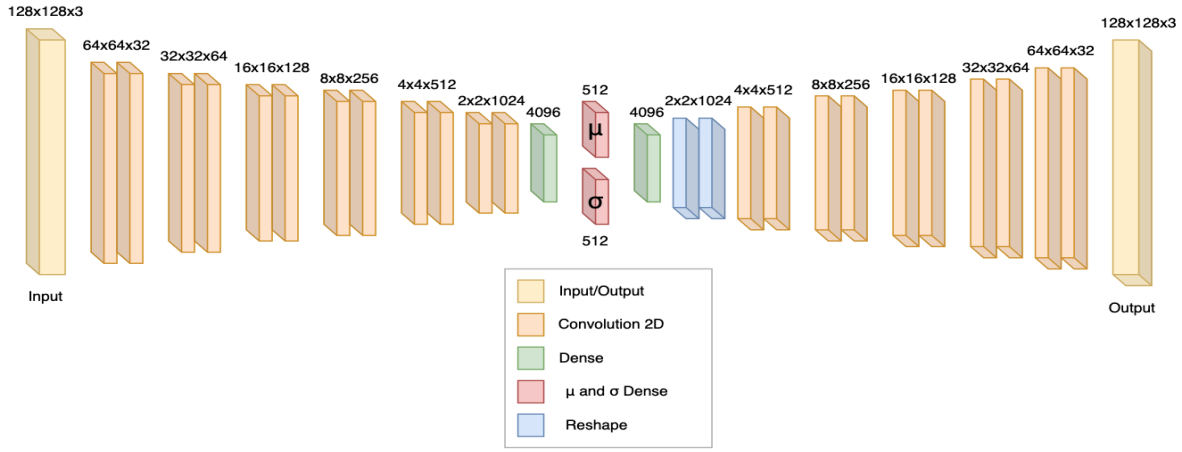

Figure 3. Variational Autoencoder Model.

The variational autoencoder (VAE) implemented in this study, as shown in Figure 3, utilizes a symmetrical design for the encoder and decoder with carefully chosen convolutional layers. The encoder's convolutional layers employ strides of 3 and a kernel size of 5 with the "swish" activation function, effectively compressing the input 128x128 RGB (Red, Green, Blue) Pokémon image into a latent space. The "tanh" activation function is utilized in the final dense layer of the encoder, centering the output around zero to facilitate efficient backpropagation, as suggested by Nwankpa et al. [8]. The decoder, employing "swish" activation with strides of 2 and kernel sizes of 3, reconstructs the image from the latent space, ensuring pixel values are bound between 0 and 1 with a sigmoid activation in the output layer.

A pivotal innovation in our VAE is the application of the reparameterization trick during the sampling phase, achieved by a Lambda layer in Keras. This layer enables the stochastic sampling of a latent vector z from a normal distribution defined by the encoded means (μ) and log variances (σ), allowing for gradient-based optimization through stochastic nodes. This key innovation enables the model to backpropagate through uncertainties and learn probabilistic mapping from input to latent space [5]. The decoder's role is to then reconstruct the input image from z, with the reconstruction fidelity assessed by mean squared error (MSE), ensuring the VAE focuses on accurately reproducing the original data.

The loss function of our VAE is augmented by the Kullback–Leibler (KL) divergence to regularize the latent space, enforcing a standard Gaussian distribution. This divergence serves as a complexity penalty, promoting the discovery of efficient data representations within the latent space. The β parameter provides a means to balance the reconstruction loss against the KL divergence, dictating the importance of latent space continuity against reconstruction precision.

The overall architecture draws inspiration from the Pixel VQ-VAE presented by Saravanan and Guzdial (2020) but lacks the vector quantization aspect, instead adopting "swish" activated convolutional blocks [12]. This strategic design choice aligns our model with traditional convolutional autoencoders, bypassing the pixel-sight blocks used in the Pixel VQ-VAE. The result is a VAE that encodes the RGB image into 512-dimensional latent vectors of means and

variances, from which the decoder efficiently reconstructs the image, demonstrating the model's robust learning and generalization capabilities.

Latent Diffusion

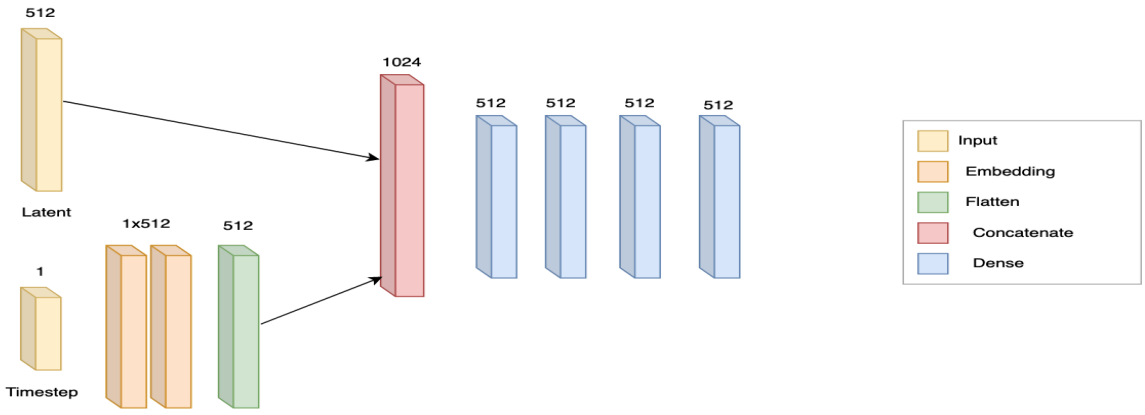

Figure 4. Latent Diffusion Denoising Probabilistic Model.

The latent diffusion model illustrated in Figure 4 represents an advanced stage in the evolution of generative models, serving as an intermediary between the encoder's compressed representations and the final image reconstructions. At its core, the model commences with the mean latent vector output by the variational autoencoder's encoder. This vector undergoes a temporal transformation through a series of embedded timestamps. Each timestamp is embedded into a higher-dimensional space and subsequently flattened to ensure compatibility with the latent vector's dimensions. The resulting concatenated layer then feeds into a multi-layer perceptron (MLP) architecture, comprised of four dense layers, all activated by the “swish” function, except for the last layer which remains linear. This design choice aims to reconstruct the latent dimensionality back to 512 units, maintaining the integrity of the encoded information.

The architecture draws inspiration from Ho et al.'s seminal work on denoising probabilistic models, which proposes a reverse process model for generative tasks [3]. While Ho et al.'s approach incorporates an attention mechanism to refine the model's focus during training and sampling phases, our implementation simplifies the process by excluding attention mechanisms, thus optimizing computational efficiency without significantly compromising the model's performance.

The absence of an activation function in the output layer preserves the raw predictions, which is a standard practice in regression tasks or when the subsequent process (such as a decoder) requires the untransformed output.

Our model, devoid of an attention mechanism, relies on the structured progression through the MLP to refine the latent vectors progressively. This refinement process mirrors the principles of diffusion models—gradually denoising the latent space to generate coherent and high-fidelity reconstructions of the original data, reflecting an understanding of the data's underlying distribution.

Evaluation

Initially, we hoped to develop a regular Autoencoder for latent diffusion, but the results were not impressive:

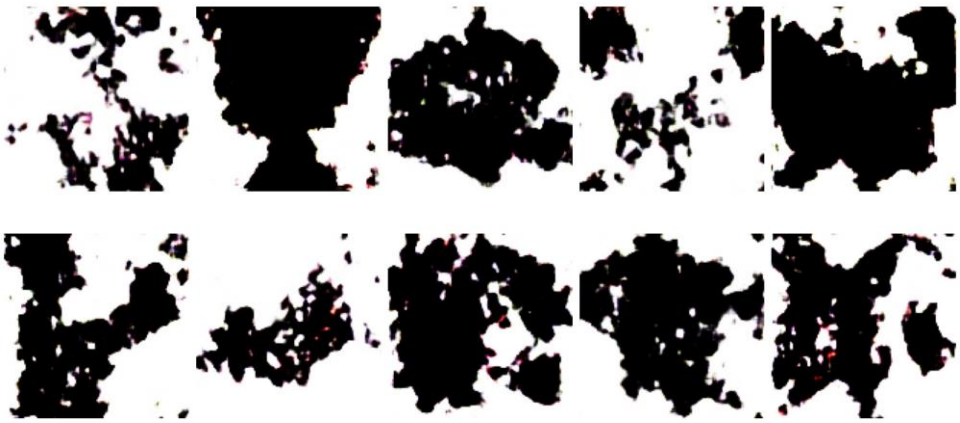

Figure 5. Latent Diffusion with Autoencoder Resulting 128x128 Pokémon Images

The reverse process model clearly had some difficulty learning the latent distribution from the Autoencoder. As a result of many such experiments, we decided to implement a Variational Autoencoder, in the hope of creating a latent distribution that would be easier for the reverse process model to learn. Indeed, this approach was clearly better:

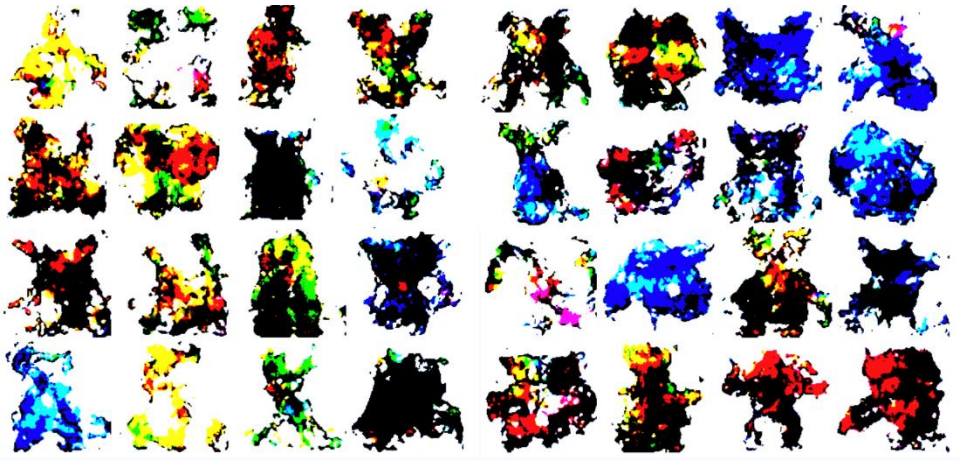

Figure 6. Latent Diffusion to VAE Decoder Resulting 128x128 Pokémon Images.

It appears that the reverse process model learned the distribution well enough that the latent vectors it produced could be decoded into colorful, although still noisy, sprites. Unfortunately, the Variational Autoencoder did not actually learn to blend the structure of the images, as Figure 7 shows. In between two latent vectors, partially drawn sprites are overlaid, which results in a noisy representation. Figure 6 shows the intersections of multiple partial sprites, unmodified in structure. The model is not powerful enough to create novel sprites by combining the semantics of existing sprites.

Figure 7. Interpolation of VAEs Embedding Space for 128x128 Pokémon Images.

Ultimately, our model met our expectations, which we kept realistic, as generating novel images with a limited amount of input data is a very complex problem. While our generated sprites are not able to represent new Pokémon-style creatures, they do contain unique features among themselves which could potentially be used as inspiration for artists to base the creation of new Pokémon on.

Conclusions and Future Work

Building on the foundational work of using latent diffusion models and variational autoencoders for Pokémon generation, future efforts will aim to refine and expand the models' capabilities. Enhancing the complexity of the latent diffusion model and improving the VAE could lead to higher quality and more diverse Pokémon creations. Exploring combined loss methods and novel activation functions may optimize model performance further. As with all deep learning models, more quality data would also be welcome. This could be achieved by developing new preprocessing techniques to make use of additional Pokémon images, not only those from the handheld games, as well as through data augmentation: by rotating and slightly deforming existing sprites. This could teach the model that the structure need not be so rigidly fixed.

Another option to explore would be a U-Net architecture, as in Rombach et al [11]. Following that paper's example would also open the door to text-to-image generation, allowing players to steer the sprite generation in terms of desired form and element types.

Ultimately our goal is integrating generated Pokémon into Pokémon Showdown (a gaming platform), providing practical application and community engagement. Additionally, mapping the latent space to understand Pokémon element types and forms could enable generation based on specific criteria, enriching the diversity and specificity of output. This focused direction promises not only to advance the generative models, but also to contribute to the rich Pokémon community and generative modeling research.

References

1. *Cs.nyu.edu*, Frey Faces Dataset. Department of Computer Science NYU, 2024, www.cs.nyu.edu/~roweis/data/frey_rawface.mat.
2. Chu, J. J., Shan, Z., & Sun, X. (2023, February). Attribute prediction of Pokémon images based on convolutional neural network. In *2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)* (pp. 129-134). IEEE.
3. Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. arXiv:2006.11239.
4. Horsley, L., & Perez-Liebana, D. (2017, August). Building an automatic sprite generator with deep convolutional generative adversarial networks. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)* (pp. 134-141). IEEE.
5. Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. arXiv:1312.6114.
6. Luo, Guangcong et al. pokemon-showdown (2020). *GitHub*. Retrieved from https://github.com/smogon/pokemon-showdown.
7. Manakov, I., Rohm, M., and Tresp, V. (2019). "Walking the tightrope: An investigation of the convolutional autoencoder bottleneck." *arXiv preprint arXiv:1911.07460*. https://arxiv.org/pdf/1911.07460.pdf
8. Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation Functions: Comparison of trends in Practice and Research for Deep Learning. arXiv:1811.03378.
9. Pérez, Luis A, et al. "Diffusion Variational Autoencoders." ArXiv (Cornell University), 1 July 2020, https://doi.org/10.24963/ijcai.2020/375. Accessed 20 Mar. 2024.
10. PokeAPI, "Sprites dataset," *GitHub*, [Online]. Available: https://github.com/PokeAPI/sprites.
11. Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022.
12. Saravanan, A., & Guzdial, M. (2020). PixelVQ: An Autoencoder for Image Generation and Editing. arXiv:2012.02801.