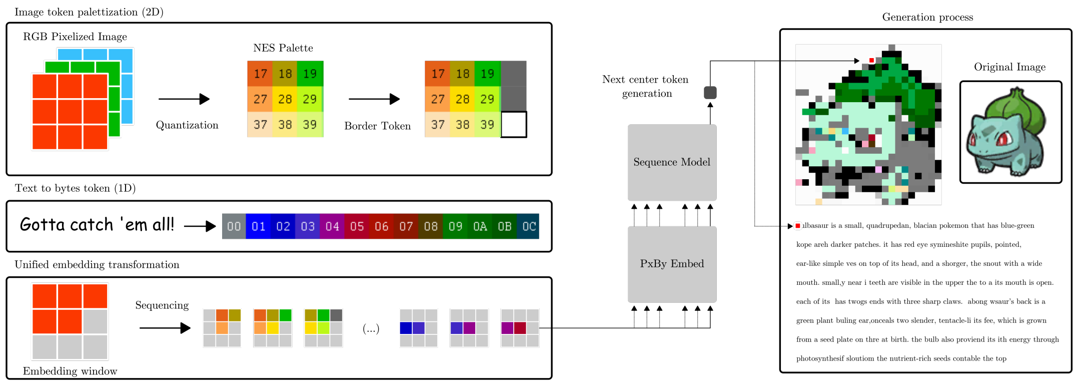
PixelBytes: Catching Unified Embedding for Multimodal Generation

Fabien Furfaro [*]

2024

Abstract

This report introduces PixelBytes Embedding, a novel approach for unified multimodal representation learning. Our method captures diverse inputs in a single, cohesive representation, enabling emergent properties for multimodal sequence generation, particularly for text and pixelated images. Inspired by state-of-the-art sequence models such as Image Transformers, PixelCNN, and Mamba-Bytes, PixelBytes aims to address the challenges of integrating different data types. We explore various model architectures, including Recurrent Neural Networks (RNNs), State Space Models (SSMs), and Attention-based models, focusing on bidirectional processing and our innovative PxBy embedding technique. Our experiments, conducted on a specialized PixelBytes Pokémon dataset, demonstrate that bidirectional sequence models with PxBy embedding and convolutional layers can generate coherent multimodal sequences. This work contributes to the advancement of integrated AI models capable of understanding and generating multimodal data in a unified manner. Code is available at `https://github.com/fabienfrfr/PixelBytes`.



ulbasaur is a small, quadrupedan, blacian pokemon that has blue-green kope areh darker patches. it has red eye symineshite pupils, pointed, ear-like simple ves on top of its head, and a shorger, the snout with a wide mouth. small,y near i teeth are visible in the upper the to a its mouth is open. each of its has twogs ends with three sharp claws. abong wsaur's back is a green plant buling ear,onceals two slender, tentacle-li its fee, which is grown from a seed plate on thre at birth. the bulb also proviend its ith energy through photosynthesif sloutiom the nutrient-rich seeds contable the top

Figure 1: PixelBytes Pokémon generation process. Left: Image and text tokenization with embedding window for modality unification. Center: Sequence generation using PxBy Embedding. Right: Generated pixelated image and text description. Red box shows a possible generation example. Note: The model currently has some inaccuracies in image generation and may invent words, indicating areas for future improvement.

1 Introduction

Recent advancements in artificial intelligence have led to increasingly generalist models, not by combining multiple specialized components, but by giving simple tasks to models where emergent properties—complex behaviors that arise from simpler underlying rules—appear. This is the case with generative language models like GPT [3]. However, these models are limited by their focus on language alone, failing to capture the full complexity of multimodal understanding [6]. To address this limitation, researchers have explored combining LLMs with other modalities [7]. But this brings us back to the initial problem, as it often results in specialized model combinations without allowing for new emergent properties. We propose "PixelBytes Embedding", a novel approach enabling unified training across modalities by capturing diverse inputs in a single, cohesive representation.

Multimodal sequence generation, which involves the creation of coherent outputs combining various data types such as text, images, and numerical sequences, presents a significant challenge in artificial intelligence [1]. While models like GPT have excelled in text generation [3], there's a growing need for unified approaches that can seamlessly handle varied data types. Building upon these findings, PixelBytes aims to address the challenge of unified text and image generation by proposing a model capable of producing mixed sequences of text and images in a coherent and unified manner. We draw inspiration from state-of-the-art sequence models, including Image Transformers [9], PixelCNN [14], and recent developments in bytes generation by mamba architectures [18].

Our research explores various architectures including Recurrent Neural Networks (RNNs), State Space Models (SSMs) [4], and Attention-based models, focusing on the effectiveness of bidirectional processing [19], novel embedding techniques (particularly PxBy embedding, which unifies pixel and byte-level representations), the impact of

[*] Corresponding author : Fabien Furfaro (`fabien.furfaro@gmail.com`).

convolutional layers, effects of model depth, input dimensionality, and size. Our experiments reveal that bidirectional sequence models with PxBy embedding and convolutional layers demonstrate good performance in generating coherent sequences.

Our results show that it is possible to train sequence models with unified embeddings for both text and pixel completion. This paper presents our methodology to construct unified sequence data from text and image, experimental results, and analysis, contributing to the development of our sequence generation embedding models. The proposed PixelBytes Embedding approach may offer valuable insights for the development of multimodal AI, potentially leading to models that can better understand and generate various types of data.

2 Model Architecture

2.1 Overview

The PixelBytes architecture is designed to seamlessly integrate multimodal data for unified sequence generation. At its core, the model incorporates two key innovative components: a specialized tokenizer sequence constructor and a unified multimodal embedding technique called PxByEmbed. The tokenizer sequence constructor is engineered to process both pixelated images and text at a byte level, enabling a consistent representation across modalities. This is complemented by PxByEmbed, our novel embedding approach that creates a unified representation for both pixel and byte data in a single, coherent space. This architecture draws inspiration from recent advancements in multimodal learning [1] and efficient sequence modeling [16], allowing PixelBytes to capture intrinsic relationships between visual and textual information effectively.

2.2 Dataset Construction

Image captioning datasets, while combining visual and textual modalities, prove unsuitable for joint text and image generation due to limited text content and difficulties in interpreting pixelated versions of high-resolution images. To address these limitations, we developed a specialized Pokémon dataset, offering advantages such as a long-standing franchise history, pixelated designs, rich descriptive text, and over 1000 unique Pokémon.

We constructed our dataset by web scraping Pokémon miniatures and descriptions from Pokepedia using Beautiful Soup [10], maintaining a 2/3 text to 1/3 image ratio. For image quantization, we employed a 55-color palette inspired by the NES, creating tokens representing different color combinations. This approach translates visual information into a format suitable for sequence modeling. The quantization and pixelization process used OpenCV and scikit-image [2, 15]. To ensure balanced representation, we adjusted the number of image and text tokens to 113 characters for each modality, allowing the model to learn equally from both visual and textual information. The final dataset, balancing text and pixelated images, is available on the Hugging Face Datasets Hub for reproducibility at `https://huggingface.co/datasets/ffurfaro/PixelBytes-Pokemon`.

3 Multimodal Embedding Algorithm

3.1 PxByEmbed: Multimodal Embedding Algorithm

At the core of our approach is the PxByEmbed algorithm, which represents mixed sequences of text and pixelated images in a unified manner. This algorithm builds upon existing embedding techniques by incorporating spatial adaptivity, allowing for representation of both textual and visual information. PxByEmbed is designed to address the specific needs of pixel-level image representation and byte-level text encoding. The algorithm uses a learned embedding matrix to map each token (text or image) to a vector space, while maintaining spatial relationships for image tokens. This approach allows our model to handle both modalities within a single framework.

3.2 Managing Transitions

PixelBytes uses a method to handle transitions between text and image tokens during sequence generation. This approach is used in both dataset construction and sequence generation (Figure 1).

For dataset construction, we use a 2D input sequence method with a 3x3 context window around each token. We use special tokens to mark transitions between text and images, and add padding to keep context sizes consistent. For text, only previous tokens are included in the context windows. The `input_seq_construct` function implements this process.

During sequence generation, the `_process_token` function manages transitions. It handles text tokens (bytes) and image tokens (tuples) differently, and takes care of special characters like newlines and tabs. A `sequence_clock` keeps track of the generation progress. Each new token's context is represented in a 3x3 matrix, which helps maintain coherence when switching between text and images.

```
Algorithm 1 PxByEmbed: Multimodal Embedding Algorithm (k=3)
Input: $V$: vocabulary size, $D$: embedding dimension
Output: Embedded representation $\mathbf{E} \in \mathbb{R}^{B \times L \times D}$
Note: $\mathbf{X}_{emb} \in \mathbb{R}^{B \cdot L \times E_{int} \times k \times k}$, $\mathbf{X}_{flat} \in \mathbb{R}^{B \cdot L \times E_{int} k^2}$, $\mathbf{X}_{proj} \in \mathbb{R}^{B \cdot L \times D}$
  Initialize:
  $k \leftarrow 3$
  $E_{int} \leftarrow \max(9, \lfloor D/k^2 \rfloor)$
  $\alpha \in \mathbb{R}^{1 \times 1 \times k \times k}$
  $\mathbf{W}_{emb} \in \mathbb{R}^{V \times E_{int}}$
  $\mathbf{W}_{proj} \in \mathbb{R}^{E_{int} k^2 \times D}$
  $\mathbf{W}_{patch} \in \mathbb{R}^{E_{int} \times E_{int} \times k \times k}$
  function PxByEmbed($\mathbf{X} \in \mathbb{Z}^{B \times L \times k \times k}$)
      $\mathbf{X}_{emb} \leftarrow \text{Permute}(\text{Embed}(\mathbf{X}, \mathbf{W}_{emb}), [0, 3, 1, 2])$
      $\mathbf{X}_{patch} \leftarrow \text{Conv2D}(\mathbf{X}_{emb}, \mathbf{W}_{patch}, \text{padding} = 1)$
      $\mathbf{X}_{combined} \leftarrow \sigma(\alpha) \odot \mathbf{X}_{emb} + (1 - \sigma(\alpha)) \odot \mathbf{X}_{patch}$
      $\mathbf{X}_{flat} \leftarrow \text{Flatten}(\mathbf{X}_{combined})$
      $\mathbf{X}_{proj} \leftarrow \mathbf{X}_{flat} \mathbf{W}_{proj}$
      $\mathbf{E} \leftarrow \text{LayerNorm}(\mathbf{X}_{proj})$
      $\mathbf{E} \leftarrow \text{Reshape}(\mathbf{E}, [B, L, D])$
      return $\mathbf{E}$
  end function
```

4 Training and Evaluation

4.1 Model Architectures

We evaluated three compact model architectures, each with fewer than 100,000 parameters: a Recurrent Neural Network (RNN) using bidirectional Long Short-Term Memory (LSTM) units [11], a Transformer [16], and Mamba, based on the State Space Model (SSM) [4]. These models were adapted to process our dataset of pixel data and bytecode sequences. Training was conducted on Kaggle using dual T4 GPUs, with a batch size of 32, sequence length of 256, stride size of 32, and learning rate of 0.001. We trained for 200 epochs, evaluating performance every 5 epochs. The resulting models are available at `https://huggingface.co/ffurfaro/PixelBytes-Pokemon`. Additionally, we developed a more specialized model for generation tasks, which is not presented in this paper.

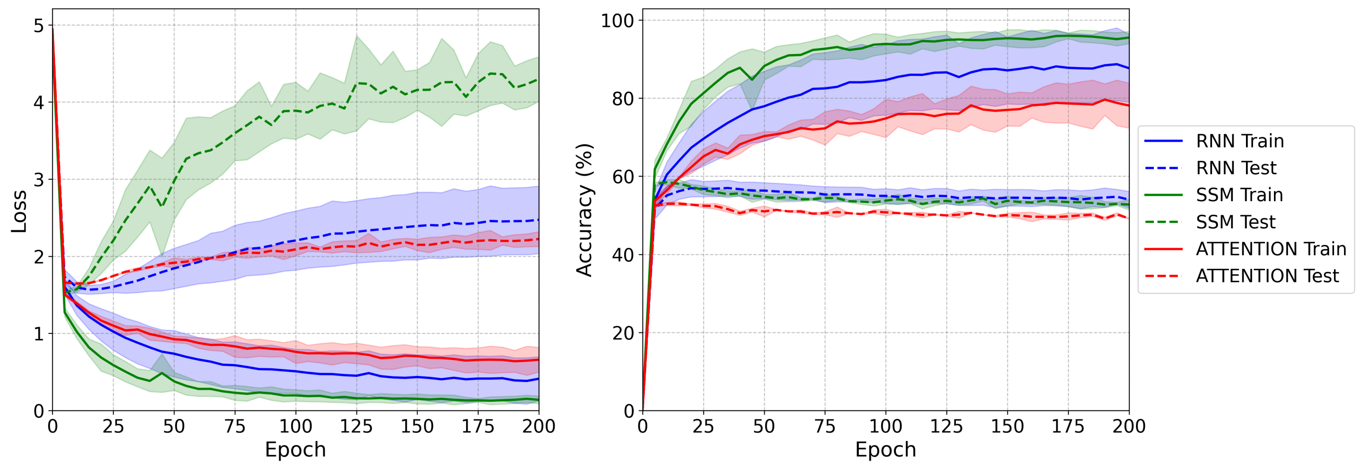


Figure 2: Training and validation metrics (Loss, Accuracy) for RNN, Transformer, and SSM models over 200 epochs.

Figure 2 shows the training and validation metrics for our three model types over 200 epochs. The State Space Models (SSM) achieved the best scores for loss and accuracy. However, the widening gap between their training and validation curves suggests they may be overfitting, meaning they learned the training data well but struggled to generalize to new examples.

In contrast, the LSTMs (referred to as RNNs in our analysis) demonstrated more balanced performance. The closer alignment of their training and validation curves indicates they may generalize better to unseen data. The Transformer model had the lowest performance among the three. This could be due to the absence of positional encoding trick, which is important for Transformers in sequence tasks.

Overall, these results highlight the strengths and weaknesses of each model type. While SSMs showed strong training performance, their tendency to overfit suggests that additional regularization techniques might be needed.

The stable performance of RNNs indicates their suitability for our task, as they balance well between fitting the training data and generalizing to new data. The Transformer's performance suggests it may require further adjustments to improve its effectiveness for this task.

4.2 Generation Evaluation Metrics

We tested different model types including State Space Models (SSM), Attention models (Att), and Recurrent Neural Networks (RNN) for generating 16 consecutive sequences. We used three metrics to evaluate their performance: Hamming Distance, Cosine Similarity, and BLEU Score [5, 8, 13].

Type	Dir.	Emb.	Conv.	In Emb	Hidden State	Depth	Hamming	Cosine	BLEU
SSM	Bi	PxBy	Y	81	64	1	0.170 ± 0.086	0.883 ± 0.105	0.753 ± 0.115
SSM	Bi	PxBy	Y	81	64	2	0.158 ± 0.074	0.896 ± 0.095	0.771 ± 0.097
SSM	Uni	PxBy	Y	81	64	2	0.166 ± 0.081	0.886 ± 0.102	0.760 ± 0.106
Att	-	PxBy	Y	81	64	1	0.157 ± 0.064	0.887 ± 0.103	0.765 ± 0.088
Att	-	PxBy	Y	81	64	2	0.159 ± 0.066	0.887 ± 0.103	0.760 ± 0.092
RNN	Bi	Center	N	81	64	2	0.185 ± 0.074	0.888 ± 0.083	0.750 ± 0.093
RNN	Bi	PxBy	Y	81	64	2	0.153 ± 0.061	0.902 ± 0.090	0.777 ± 0.083
RNN	Bi	PxBy	Y	162	64	2	0.152 ± 0.062	0.905 ± 0.089	0.778 ± 0.084
RNN	Bi	PxBy	Y	36	64	2	0.152 ± 0.061	0.904 ± 0.090	0.778 ± 0.083
RNN	Bi	PxBy	Y	81	128	2	0.153 ± 0.063	0.903 ± 0.091	0.776 ± 0.086
RNN	Bi	PxBy	Y	81	32	2	**0.149** ± 0.062	0.899 ± 0.095	**0.785** ± 0.082
RNN	Bi	PxBy	Y	81	64	1	0.149 ± 0.062	0.897 ± 0.096	0.780 ± 0.085
RNN	Bi	PxBy	Y	81	64	3	0.153 ± 0.063	**0.906** ± 0.087	0.776 ± 0.086
RNN	Bi	PxBy	N	81	64	2	0.151 ± 0.062	0.903 ± 0.090	0.779 ± 0.084
RNN	Uni	PxBy	Y	81	64	2	0.153 ± 0.064	0.904 ± 0.088	0.777 ± 0.087

Table 1: Comparison of model characteristics and performance (mean ± std)

The results show that RNN models with PxBy embedding generally perform better than other types. The best Hamming distance (0.149 ± 0.062) and BLEU score (0.785 ± 0.082) come from the bidirectional PxBy RNN model with convolution, 81 embedding dimensions, and 32 hidden state dimensions. This suggests that a smaller hidden state in the RNN model can effectively capture sequence patterns.

The highest cosine similarity (0.906 ± 0.087) is from the 3-layer RNN model, indicating that more layers can improve sequence alignment. Adding convolution usually improves RNN model performance. Interestingly, changing the embedding dimension (36, 81, 162) in RNN models didn't affect performance much, suggesting these models can handle different embedding sizes well. The center embedding in RNNs didn't perform as well as PxBy embedding for this task.

5 Results and Discussion

Our experiments with various model architectures tested the effectiveness of unified text and image generation. Bidirectional RNN models using PixelBytes (PxBy) embedding with convolutional layers generally performed better across our metrics. The PxBy embedding improved the mean BLEU score from 0.750 to 0.777 compared to a center-only approach, indicating its effectiveness in representing multimodal data. This finding aligns with recent work emphasizing the importance of effective multimodal embeddings in text-to-image generation tasks [17].

Bidirectional models showed a slight advantage over unidirectional ones, with BLEU scores of 0.777 and 0.776 respectively. This suggests that bidirectionality may help capture sequential dependencies, consistent with other studies in sequence modeling [12]. Notably, varying state dimensions (32, 64, 128) had minimal impact on performance, with BLEU scores ranging from 0.775 to 0.784. While RNN models showed strong performance in our experiments, State Space Models (SSM) demonstrated rapid convergence during training, suggesting potential for future exploration in multimodal data processing. This is in line with recent advancements in SSM for sequence modeling tasks [4].

6 Conclusion and Future Work

This study on PixelBytes highlights the potential for unified text-image generation, contributing to the growing field of multimodal AI [1]. The performance of bidirectional RNN models with PxBy embedding underscores the importance of capturing both spatial and contextual information in multimodal data. Future work could focus on refining the PxByEmbed algorithm, testing on larger and more diverse datasets, exploring creative applications of multimodal generation, and further optimizing SSM and Attention models. These efforts aim to advance multimodal sequence modeling and generation techniques.

References

[1] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.

[2] Gary Bradski. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000.

[3] Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[4] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

[5] Richard W Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[6] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Barun Patra, et al. Language is not all you need: Aligning perception with language models. *Advances in Neural Information Processing Systems*, 36:72096–72109, 2023.

[7] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[8] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[9] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pages 4055–4064. PMLR, 2018.

[10] Leonard Richardson. Beautiful soup documentation, 2007.

[11] Jürgen Schmidhuber, Sepp Hochreiter, et al. Long short-term memory. *Neural Comput*, 9(8):1735–1780, 1997.

[12] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[13] I Sutskever. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[14] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

[15] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[16] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[17] Anita L Verő and Ann Copestake. Efficient multi-modal embeddings from structured data. *arXiv preprint arXiv:2110.02577*, 2021.

[18] Junxiong Wang, Tushaar Gangavarapu, Jing Nathan Yan, and Alexander M Rush. Mambabyte: Token-free selective state space model. *arXiv preprint arXiv:2401.13660*, 2024.

[19] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.