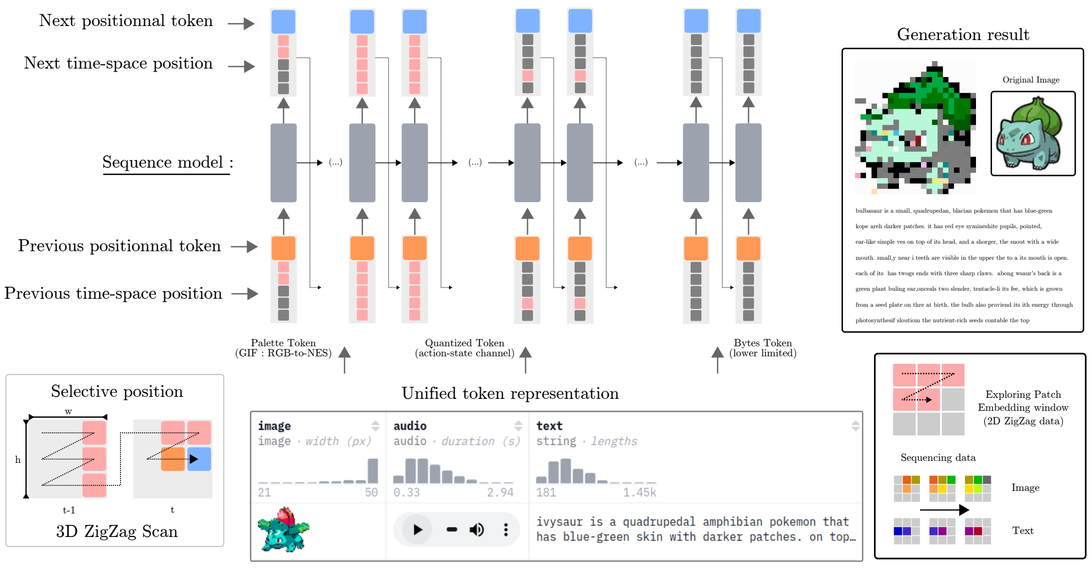
PixelBytes: Catching Unified Representation for Multimodal Generation

Fabien Furfaro*

2024

Abstract

This report presents PixelBytes, an approach for unified multimodal representation learning. Drawing inspiration from sequence models like Image Transformers, PixelCNN, and Mamba-Bytes, we explore integrating text, audio, action-state, and pixelated images (sprites) into a cohesive representation. We conducted experiments on a PixelBytes Pokémon dataset and an Optimal-Control dataset. Our investigation covered various model architectures, including Recurrent Neural Networks (RNNs), State Space Models (SSMs), and Attention-based models, with a focus on bidirectional processing and our PxBy embedding technique. We evaluated models based on data reduction strategies and autoregressive learning, specifically examining Long Short-Term Memory (LSTM) networks in predictive and autoregressive modes. Our results indicate that autoregressive models perform better than predictive models in this context. Additionally, we found that diffusion models can be applied to control problems and parallelized generation. PixelBytes aims to contribute to the development of foundation models for multimodal data processing and generation. The project's code, models, and datasets are available online [10].



ivysaur is a quadrupedal amphibian pokemon that has blue-green skin with darker patches. on top…

bulbasaur 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: Overview of the PixelBytes approach: (Left) Scanning process reads different modalities at specific positions. (Center) Example of our dataset and autoregressive generation across modalities. (Right) Generation example, with PxBy embedding window displayed below. Note: The model currently shows some inaccuracies in image generation and may produce incorrect 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 (like Gato from DeepMind [29]), but by assigning simple tasks to models where emergent properties—complex behaviors arising from simpler underlying rules—appear. This is exemplified by generative language models such as GPT [7]. However, these models are constrained by their focus on language alone, failing to capture the full complexity of multimodal understanding [16]. To address this limitation, researchers have explored integrating Large Language Models (LLMs) with other modalities [25]. However, this approach often results in specialized model combinations without fostering new emergent properties. We propose "PixelBytes", a novel approach enabling unified training across modalities by representing diverse inputs in a single, cohesive format.

Multimodal sequence generation, which involves creating coherent outputs combining various data types such as text, images, and numerical sequences, presents a significant challenge in artificial intelligence [2]. While models

*fabien.furfaro@capgemini.com

like GPT have excelled in text generation [7], there is a growing need for unified approaches that can seamlessly handle diverse data types. Building on these findings, PixelBytes addresses the challenge of unified text, audio, and image generation by proposing a model capable of producing coherent mixed sequences of text, audio, and images. Our approach is inspired by state-of-the-art sequence models, including Image Transformers [28], PixelCNN [35], and recent developments in byte generation by Mamba architectures [39].

Our research investigates various architectures including Recurrent Neural Networks (RNNs), State Space Models (SSMs) [8], and Attention-based models, examining the effectiveness of bidirectional processing [23, 32, 42], innovative embedding techniques (particularly PxBy embedding, which integrates pixel and byte-level representations), the influence of convolutional layers, and the effects of model depth, input dimensionality, and size.

Our experiments, conducted on a specialized PixelBytes Pokémon dataset, suggest that autoregressive models with balanced data reduction strategies perform better than predictive models in terms of accuracy and loss. We initially explored bidirectional sequence models with PxBy embedding and convolutional layers, but our final results focus on Long Short-Term Memory (LSTM) networks in both predictive and autoregressive modes. Additionally, we tested our approach on control problems using diffusion models. This paper presents our method for constructing unified sequence data from text, audio, action-state, and pixelated images (sprites), along with our experimental findings and analysis. Through a flexible approach to multimodal modeling, PixelBytes seeks to contribute to the development of foundation models capable of processing and generating multimodal data.

2 Exploration for a Unified Representation

2.1 Hypothesis Testing Framework

The quest for a unified data representation across different modalities presents significant challenges. Text data typically exhibits a one-dimensional dependency on preceding words with discrete values [4]. Audio signals have a temporal dimension with continuous values [34]. Action-state representations in robotics can be similar to audio but with correlations between channels [5]. Images and animations combine spatial and temporal dimensions across discrete RGB channels [19]. Given these diverse characteristics, many researchers have focused on combining separate embedding models into a single framework, as seen in projects like ImageBind [11] and RT-2 [43], rather than seeking a truly unified data representation. However, to build a model that comprehensively understands different modalities without intermediate alignment steps, exploring a unified data representation becomes necessary. In this section, we will examine several hypotheses:

- Can we quantize data such that each element becomes a token? [41]
- Is predicting only the next value sufficient for a sequence model to learn effectively?
- For dimensions higher than one, is applying a convolutional filter necessary?
- For space-time dependency, what is the importance of bidirectionality in models?

Through these investigations, we aim to explore a method of representing data that could enable models to understand various types of modalities.

2.2 Conceptual Multimodal Embedding

2.2.1 Dataset Construction

To evaluate our hypotheses on unified representation, we required a dataset combining visual and textual data suitable for byte-level processing. Image captioning datasets proved inadequate due to limited text content and challenges in interpreting pixelated versions of high-resolution images. Consequently, we created a specialized Pokémon dataset, offering pixelated designs and rich descriptive text. Data was collected by web scraping Pokémon miniatures and descriptions from Pokepedia using Beautiful Soup [30], maintaining a 2/3 text to 1/3 image ratio. For image processing, we utilized a 55-color palette inspired by the NES, creating tokens for various color combinations. This approach enabled us to represent visual information in a format compatible with our tokenizer.

To manage transitions between text and image tokens, we developed a 2D input sequence method utilizing a 3x3 context window around each token with a 2D zigzag scheme (Figure 1). Special tokens denote transitions between text and images, with padding added to maintain consistent context sizes. The padding value is 0, line transition value is 1, and modality transition value is 2. For text, only preceding tokens are included in the context windows. Employing OpenCV and scikit-image [6, 36] for image quantization and pixelization, we adjusted all entries to have 113 indices, balancing text and image tokens. The resulting dataset, combining text and pixelated images, is available on the Hugging Face Datasets Hub [10] for reproducibility. It includes a "pixelbyte" column for this specific data representation.

2.2.2 Embedding Techniques

Our exploration of unified representation begins with integrating image-text data for sequence generation. We developed a tokenizer that processes the "pixelbyte" column. This is paired with an embedding technique called PxByEmbed, which creates a unified representation for pixel and byte data in a single space. PxByEmbed is designed to test our hypotheses about the effectiveness of convolutional filters for higher-dimensional data. It uses a learned embedding matrix to map each token (text or image) to a vector space, while maintaining spatial relationships for image tokens. PxByEmbed incorporates a simple convolutional layer and an adaptive mixing mechanism. This design allows us to investigate whether predicting only the next value (with or without only previous value) is sufficient for effective learning in a sequence model.

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 E
end function
```

The algorithm processes input sequences in 3x3 patches, embedding each element and then applying a learnable convolution. The embedding pad value is set to 0 to avoid influencing the training of subsequent tokens. These two representations are then combined using a learned parameter, allowing the model to adaptively balance between local and global information.

2.3 Model Architectures Evaluated

We evaluated three compact model architectures: a Recurrent Neural Network (RNN) using Long Short-Term Memory (LSTM) units [31], a Transformer [37], and a State Space Model (SSM) based on Mamba [12]. For both the RNN and Mamba architectures, we compared variants with and without a bidirectional first layer. Each model was constrained to fewer than 100,000 parameters and adapted to process our dataset of pixel data and bytecode sequences. The models were trained on Kaggle using T4 GPUs, with a batch size of 32, sequence length of 256, and learning rate of 0.001 for 200 epochs. The trained models are available on the Hugging Face Model Hub [10].

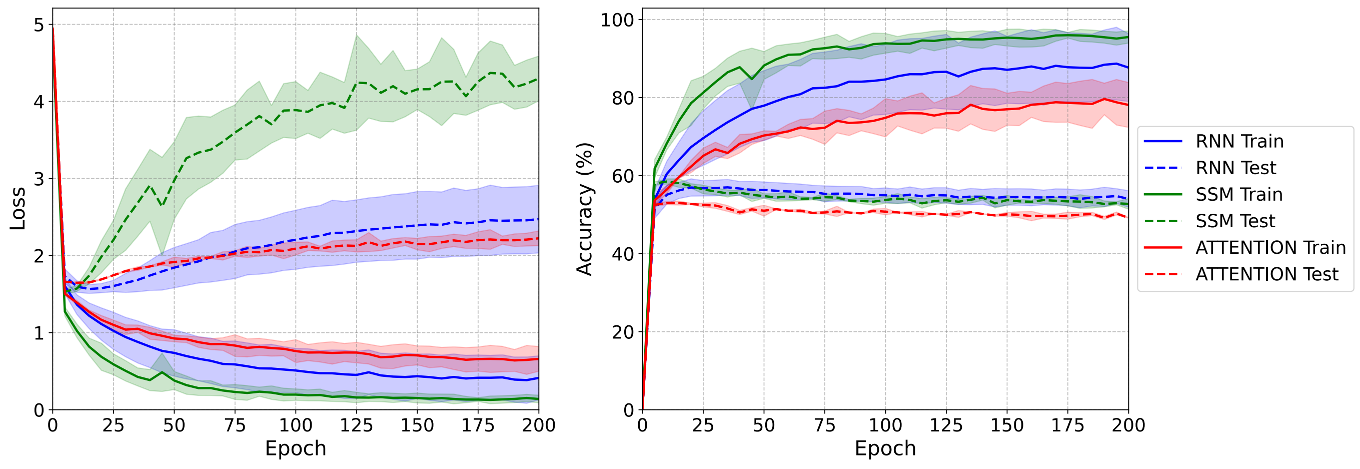


Figure 2: Training and validation metrics for RNN, Transformer, and SSM models over 200 epochs.

2.4 Comparative Analysis

Figure 2 illustrates the training and validation metrics for our three model types. The SSM achieved the best scores for loss and accuracy but exhibited signs of overfitting. The RNN demonstrated more balanced performance, suggesting better generalization to unseen data. The Transformer had the lowest performance, possibly due to the absence of positional encoding. The strong performance of the SSM supports the potential of unified representation for pixel and byte data. However, its tendency to overfit suggests that predicting only the next value might not be sufficient for effective learning in all cases. The balanced performance of the RNN indicates that simpler architectures can still be effective for our task. The Transformer's lower performance suggests that complex attention mechanisms may not always be necessary or beneficial for this type of data.

2.4.1 Generation Evaluation Metrics

To assess the effectiveness of our approach and various model architectures, we evaluated their generation capabilities. We tested State Space Models (SSM), Attention models (Att), and Recurrent Neural Networks (RNN) in generating 32 consecutive sequences. Our evaluation used three metrics: Hamming Distance [13], Cosine Similarity, and BLEU Score [27].

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 reveal variations across different model configurations. Among RNN models with PxBy embedding, we observed a Hamming distance of 0.149 ± 0.062 and a BLEU score of 0.785 ± 0.082 for a bidirectional model with convolution, 81 embedding dimensions, and 32 hidden state dimensions. The highest cosine similarity (0.906 ± 0.087) was achieved by a 3-layer RNN model. We noted some differences in performance between models with and without convolution, as well as between unidirectional and bidirectional configurations, though these differences were not always substantial. Varying the embedding dimension (36, 81, 162) in RNN models resulted in similar performance levels. RNN models using center embedding showed different results compared to those using PxBy embedding. The performance of SSM and Attention models varied in comparison to RNN models across the different metrics, but no model type consistently outperformed the others across all measures.

2.5 Identified Challenges

Our initial results revealed several limitations in our embedding approach. While we observed variations in performance across different model configurations, the differences were often not substantial [40]. The PxBy embedding, which we initially considered promising, did not consistently outperform simpler approaches across all metrics and model types. Based on these findings, we recognized the need to refine our approach. The repetition of sequences in our generated output indicated that our embedding method might not be capturing the full range of patterns in our data [3].

3 Optimizing Unified Representation

3.1 Refined Embedding Approach

To address the challenges identified in our initial experiments, we propose a revised embedding strategy. Instead of using a convolutional approach, we now focus on six specific positions within each token, with the input dimension equal to the output dimension. This adjustment allows for a larger embedding size while potentially enhancing the model's ability to capture relevant patterns. We also recognized the need for a more flexible tokenizer [20]. Our initial implementation proved cumbersome when generating new data, highlighting the importance of a more versatile approach. We are exploring methods to integrate all necessary functionality into the tokenizer itself, which should streamline our overall pipeline and potentially improve performance. These refinements reflect a shift from our initial exploration towards a more focused approach to unified representation. While our initial results provided valuable insights, they also revealed the complexities inherent in multimodal sequence modeling and the need for continuous iteration in our methods [21].

3.1.1 Dataset Construction

For our refined approach, we developed a new dataset combining images, text, and audio extracted from Pokemon sprite animations. This dataset, available on the Hugging Face Dataset Hub [10], was compiled through web scraping of Pokepedia and includes descriptions of the Pokemon along with their associated cries. The dataset comprises animated, pixelated GIFs of Pokemon sprites as the visual component. The audio files are two-channel recordings: Channel 1 contains the original mono sound of the Pokemon cry, while Channel 2 features a filtered version simulating a bits Game Boy speaker output to verify our approach for control problems. This setup enables us to model a simplified dynamic physical system, where the original sound acts as the "action" input and the filtered output represents the "state" of the system. The transfer function of this bandpass filter can be approximated as:

$$H(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{1}$$

where K is the gain, ω_n is the natural frequency, and ζ is the damping ratio. These parameters can be adjusted to closely match the frequency response of a Game Boy speaker.

3.2 Enhanced Tokenization Strategy

Building on our previous work, we developed an improved tokenization strategy using the ActionPixelBytesTokenizer. This tokenizer addresses multimodal data more effectively, including text, images, and audio, while maintaining a unified representation. It employs a combined vocabulary that includes ASCII bytes, RGB values from the NES palette, and action states for control and audio. This approach aims to create a consistent representation across different data types. For text processing, the tokenizer converts to lowercase ASCII bytes. Images are converted to Lab color space and quantized to the nearest NES palette color. Audio data is normalized and mapped to predefined action states, with the setpoint reset to zero (standard equilibrium). The token vocabulary now comprises 151 tokens, where index 0 corresponds to the null padding value, and indices 1 and 2 are transition values. A key aspect of the new tokenizer is its sequence construction method. Instead of using convolutional methods, we focus on six specific positions for each token to avoid repetition in the sequencing of a 3D zigzag scheme (Figure 1). This approach creates context-target pairs that aim to capture relationships between neighboring tokens in both space and time. The sequence construction algorithm is detailed below:

```
**Algorithm 2** Create Sequence Data
**Input:** $\mathbf{X} \in \mathbb{R}^{T\times H\times W}$: context array
**Output:** $\mathbf{C} \in \mathbb{R}^{THW\times 6}$: context, $\mathbf{Y} \in \mathbb{R}^{THW\times 1}$: targets
**Note:** $T$: time steps, $H$: height, $W$: width

  Initialize: $\mathbf{P} \in \mathbb{R}^{(T+1)\times(H+2)\times(W+2)}$ as padded array
  function CreateSequenceData($\mathbf{X}$)
      $\mathbf{P} \leftarrow$ Pad($\mathbf{X}, (1,1,1,1,1,0)$, mode='constant', value=0)
      $\mathbf{S}_1 \leftarrow \mathbf{P}_{1:T,1:H,2:W+1}$
      $\mathbf{S}_2 \leftarrow \mathbf{P}_{1:T,2:H+1,2:W+1}$
      $\mathbf{S}_3 \leftarrow \mathbf{P}_{1:T,3:H+2,2:W+1}$
      $\mathbf{S}_4 \leftarrow \mathbf{P}_{2:T+1,1:H,1:W}$
      $\mathbf{S}_5 \leftarrow \mathbf{P}_{2:T+1,2:H+1,1:W}$
      $\mathbf{S}_6 \leftarrow \mathbf{P}_{2:T+1,1:H,2:W+1}$
      $\mathbf{C} \leftarrow$ Stack($[\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4, \mathbf{S}_5, \mathbf{S}_6]$, dim=-1)
      $\mathbf{C} \leftarrow$ Reshape($\mathbf{C}, [THW, 6]$)
      $\mathbf{Y} \leftarrow$ Reshape($\mathbf{X}, [THW, 1]$)
      for $i \leftarrow 2$ to $THW$ do
          $\mathbf{C}_{i,6} \leftarrow \mathbf{Y}_{i-1,1}$
      end for
      return $\mathbf{C}, \mathbf{Y}$
  end function
```

This algorithm constructs a context array for sequence modeling. It pads the input to handle boundary conditions and extracts six slices to represent spatial and temporal relationships. The context array **C** is formed by stacking these slices, while the targets **Y** are derived from the input. The last column of **C** is set to the true previous value, implementing an autoregressive feature. The tokenizer is designed to handle various input combinations and includes functionalities such as special token handling and padding.

3.3 Autoregressive Model Architecture

Building upon our initial approach findings, we developed the aPxBySequenceModel architecture. This architecture is designed to handle both predictive and autoregressive tasks using a Long Short-Term Memory (LSTM) network. The model comprises three main components: an embedding layer, an LSTM sequence model, and a fully connected output layer. The embedding layer maps input tokens to a continuous vector space, with the embedding size calculated based on the input dimension. Specifically, the embedding size is determined by dividing the overall embedding size by the number of positions we focus on within each token. This approach aligns with our revised embedding strategy, which emphasizes six specific positions within each token (with padding 0 to avoid influencing training). The LSTM layer aims to capture temporal dependencies in the sequence data, potentially enhancing pattern recognition across different modalities. The model operates in two distinct modes:

- Predictive mode: In this configuration, the model takes six input values and attempts to predict only the next token.
- Autoregressive mode: Here, the model's output dimension matches the input dimension (Figure 1). Additionally, the output is restructured by multiplying it with the vocabulary size, enabling the model to generate sequences based on the learned representations.

During the forward pass, input data is processed through the embedding layer, then through the LSTM layers, and finally through the fully connected layer. The output shape is adjusted based on the operating mode, which may provide the flexibility we found lacking in our initial implementation.

3.3.1 Model Training and Data Management

For data management, we developed the `TokenPxByDataset` class to handle multimodal inputs, including text, image, and audio data. This class generates overlapping sequences from longer inputs, facilitating the model's capture of context across sequence boundaries. It optimizes memory usage through on-the-fly data retrieval, preparing samples only as needed. The class ensures consistent sequence lengths by implementing circular padding for sequences extending beyond an item's end. These features enable efficient processing of variable-length inputs during training.

Our training process incorporates several enhancements for efficiency and monitoring. In the `_process_epoch` function, we manage both autoregressive and non-autoregressive modes. For autoregressive mode, we reshape input and output sequences, using the input sequence as the target. In non-autoregressive mode, we flatten the outputs and use provided labels as the target. This flexibility allows the model to adapt to different tasks. The `train_model` function alternates between training and validation phases, enabling regular performance evaluation. We employ gradient accumulation to simulate larger batch sizes, beneficial when GPU memory is limited. The training loop tracks both training and validation metrics (loss and accuracy) for each epoch, saving these metrics to a CSV file. We also implement model checkpointing to retain the best model based on validation loss.

3.4 Performance Evaluation

We evaluated three LSTM models: one in predictive mode and two in autoregressive mode, each with approximately 4 million parameters. The models were trained on Kaggle using T4 GPUs. We utilized an embedding size of 128, a hidden size of 512, and two layers. Training was conducted for 100 epochs with a batch size of 32, a learning rate of 0.001, and a sequence length of 1024.

3.4.1 Results Comparison

To manage data proportions, we applied different reduction strategies for image and audio data. This was particularly important for audio data in the autoregressive mode, as it contains more null values to predict, which could potentially impact training. Table 2 presents the final performance metrics after training.

Table 2: Comparison of model performance after 100 epochs

Model	Train Loss	Train Accuracy	Val Loss	Val Accuracy
Autoregressive (2,2)	0.2211	0.9329	0.4519	0.8852
Autoregressive (4,2)	0.2346	0.9290	0.4914	0.8726
Predictive (2,2)	0.8810	0.7440	2.1144	0.6009

The numbers in parentheses indicate the reduction factors for (audio, image) data. Lower loss and higher accuracy indicate better performance. Both autoregressive models outperformed the predictive model in terms of accuracy and loss on both training and validation sets. The autoregressive model with balanced reduction (2,2) achieved slightly better results than the one with more aggressive audio reduction (4,2). Although the predictive model had lower performance overall, it still reached reasonable accuracy given the task complexity. This suggests that our pixelbyte representation is effective, especially when using autoregressive learning.

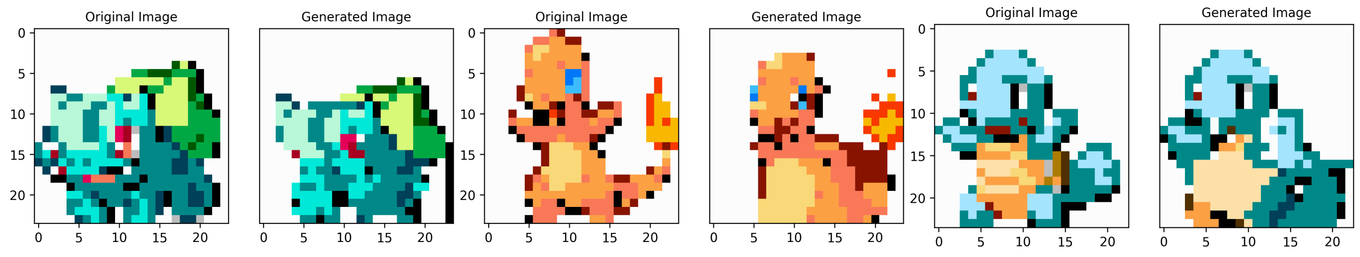


Figure 3: Generation results for the 1st Generation Starter Pokémon using the autoregressive model with a temperature of 0.1. Reference images and generated images are paired sequentially.

The complete frame generation results, as shown in Figure 3, demonstrate spatial consistency in image creation, despite the model being unidirectional. This suggests that data representation may be more important than the architecture itself, aligning with recent findings in multimodal learning [2]. The generated images show coherence in colors and sizes, indicating that the model has captured some key visual features. While our model does not match the image quality of state-of-the-art diffusion models [9], it demonstrates the capability to capture multiple modalities (image, text, audio) within a single framework. This aligns with recent trends in multimodal co-learning [24], although further optimization is needed to improve generation quality. The trained models are available on the Hugging Face Model Hub [10] for further examination and replication of our results.

4 Toward application in control problem

The auto-regressive approaches we previously explored showed promising results in image sprite generation. However, these structures may have limitations for rapid image generation and training of bidirectional models. To address these challenges, we incorporated diffusion-based models into our research, which offer more flexibility for various applications. Diffusion processes have shown good results in control optimization problems, as seen in recent work on Denoising Diffusion Probabilistic Models (DDPM) [14] and planning with diffusion [17]. In this part, we compare a semi-diffusive model with an auto-regressive model in the context of optimal control.

4.1 Dataset Optimal Control Construction

To create our optimal control dataset, we used linear systems that are stable, observable, and controllable, based on Kalman's principles [18]. We followed these steps:

1. We define the linear system using state-space matrices (A, B, C, D) as described in control theory [26]:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2}$$

   where x is the state vector, u is the input vector, and y is the output vector.

2. We check that the system meets Kalman's conditions:

   - Stability: For a transfer function $H(s) = \frac{N(s)}{D(s)}$, all eigenvalues λ_i of A and all roots p_i of $D(s)$ must satisfy:

$$\begin{cases} Re(\lambda_i) < 0 \text{ and } Re(p_i) < 0 & \text{for continuous-time systems} \\ |\lambda_i| < 1 \text{ and } |p_i| < 1 & \text{for discrete-time systems} \end{cases} \tag{3}$$

   - Controllability: $\text{rank}[B \quad AB \quad A^2B \quad \ldots \quad A^{n-1}B] = n$, where n is the number of states.
   - Observability: $\text{rank}[C \quad CA \quad CA^2 \quad \ldots \quad CA^{n-1}]^T = n$

3. For each (initial condition, setpoint) pair, we calculate the optimal control sequence using the Linear Quadratic Gaussian (LQG) approach [1]. The LQG controller minimizes:

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{4}$$

   where Q and R are weighting matrices.

We also added bang-bang controller responses with different time delays and noise levels. The bang-bang control law is:

$$u(t) = \begin{cases} u_{max}, & \text{if } e(t) > 0 \\ u_{min}, & \text{if } e(t) < 0 \end{cases} \tag{5}$$

where $e(t)$ is the error signal.

Lastly, we sampled and quantized the continuous signals to create a discrete representation for our model. The dataset is available on Hugging Face [10].

Table 3 shows the ranges of parameters used in dataset generation:

Table 3: Parameter Ranges for Dataset Generation

Parameter	**Range**
System order	1 - 5
Eigenvalues of A	[-5, -0.1]
Elements of B	[-1, 1]
Elements of C	[0, 1]
Initial conditions	[-1, 1]
Setpoints	[-1, 1]
Bang-bang delay	0 - 5 time steps
Noise standard deviation	0 - 0.1

4.2 Diffusion Model Architecture

We use a modified bidirectional LSTM to handle discretized control sequences. Our model, shown in Algorithms 3 and 4, can be used in various generation and control scenarios. The model can switch between predictive and diffusion modes, which may be useful for different types of training problems.

Algorithm 3 Forward Pass (F)
Input: x: input, t: time step, m: mask
Output: processed output
Note: E: Embedding, S: SequenceModel,
FC: FullyConnected
n_d: num_diffusion_steps, s_l: seq_len,
p_d: pxby_dim, p_e: pxby_emb, b_s: batch_size

```
procedure F($x, t, m$) $x \leftarrow E(x)$
    if $o =$ "d" then
        if $t =$ null then $t \leftarrow$ R$(n_d - 1, n_d + 1)$
        end if
        if $m =$ null then $p \leftarrow$ R$(0, s_l, 3s_l/4)$
     $m \leftarrow$ CM$(p, s_l, p_d, p_e)$
        end if
     $\alpha \leftarrow 1 - t/n_d$
     $n \leftarrow$ RN($x$.shape)
     $x \leftarrow$ W$(m = 1, x, (1 - \alpha) * n + \alpha * x)$
    end if
     $x \leftarrow S(x)$
     $x \leftarrow$ FC$(x)$
    if $o =$ "p" then return $x$
    elsereturn RS$(x, [b_s, s_l, p_d, -1])$
    end if
end procedure
```

Algorithm 4 Generation Process (G)
Input: i: input, T: temperature, m_l: max length
Output: generated sequence
Note: F: Forward, S: Softmax, M: Multinomial
R: RandomInt, RN: RandomNormal, W: Where
CM: CreateMask, RS: Reshape, SU: ScatterUpdate

```
procedure G($i, T, m_l$) $c \leftarrow i$
    for $j \leftarrow 1$ to $m_l$ do $o \leftarrow F(c)$
        if $ob =$ "p" then $p \leftarrow S(o[:, -1]/T)$
     $n \leftarrow M(p)$
     $c[:, -(j + 1)] \leftarrow n$
        else if $ob =$ "d" then $ps \leftarrow$ RP()
     $p \leftarrow S(o[:, ps]/T)$
     $t \leftarrow M(p)$
     $c \leftarrow SU(c, ps, t)$
        else$p \leftarrow S(o[:, -1]/T)$
     $n \leftarrow M(p)$
     $c \leftarrow C(c, n)$
        end if

    end for

     return $c$
end procedure
```

4.2.1 Model training and performance

We trained the models on Kaggle using T4 GPUs, with a batch size of 32, sequence length of 512 (256 overlap), 256 hidden sizes for unidirectional and 128 for bidirectional diffusion model, and learning rate of 0.001 for 200 epochs. The difference in size between the bidirectional and unidirectional versions is to avoid doubling the network weight size. During training, we faced challenges with our 6-input dataset's unbalanced nature. The '0' token is frequent and, while not in the embedding, affects the diffusion process and output generation. To address this, we used a weight normalization for the optimizer based on token frequency:

$$w'_i = w_i \cdot \frac{\sqrt{\sum_j f_j}}{\sqrt{f_i}} \tag{6}$$

where w'_i is the normalized weight, w_i is the original weight, and f_i is the frequency of token i in the dataset. We tested different model sizes to balance performance and efficiency. Table 4 shows our train and evaluation results:

Model Size	Direction	Train Type	Loss	Accuracy (%)
1.2M	Unidirectional	Auto-regressive	0.19	92.1
0.8M	Bidirectional	Diffusion	0.30	95.9

Table 4: Diffusion Model Architectures and Performance

Our results show that the bidirectional diffusion model, with fewer parameters, achieves higher accuracy than the unidirectional auto-regressive model. This suggests the diffusion-based approach may better capture control sequence complexities. The trained models are available on the Hugging Face Model Hub [10].

4.3 Setpoint following with diffusion generation

We tested our training on a simple case: a transfer function of $\frac{1}{x+1}$ with a setpoint of 0.5. We used the "Gymsetpoint" library [33], made for reinforcement learning of an LTI system with input setpoint and targeted output action. We start with a noisy bang-bang controller. After 126 time steps, we begin generating the action to reach our setpoint step-by-step. We apply a mask to N repeated action values (targets) and an incomplete mask on the expected state setpoints. We then use the average of the targets generated by diffusion for the next action step only. We generate 125 consecutive actions to reach 251 total steps. We did this experiment 100 times.

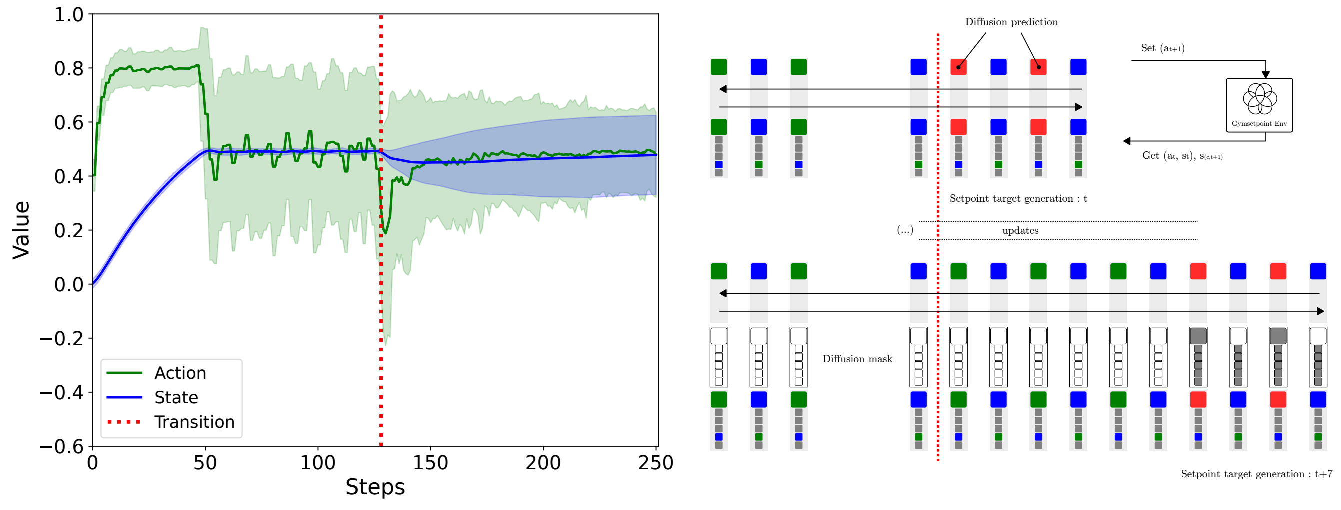


Figure 4: Generation approach for control problem. Left: generation result for linear problem. Right: generation step with setpoint and targeted diffusion action following input mask.

Figure 4 shows that the diffusion-based control is more precise and robust than bang-bang control, despite more input variability. The bang-bang control (left part of the figure) has high input variability before the diffusion-controlled region. The diffusion control works well but can sometimes produce odd results. This means we might need extra checks for real-world use. These results suggest our diffusion model could perform better than some control methods in certain cases. However, the occasional odd results show we need good error-checking and backup plans in real use, or we might need to retrain the model for specific cases.

5 Discussion and Future Directions

Our experiments with various model architectures for unified text and image generation have given us new insights and led to changes in our approach. We initially explored bidirectional RNN models using PixelBytes (PxBy) embedding with convolutional layers, expecting better multimodal data representation. However, we found limitations in this approach, leading us to reconsider our tokenizer design.

Our results with LSTM models have been informative. The autoregressive models performed significantly better than the predictive model, suggesting that keeping equal input and output dimensions is important for our task. This aligns with recent research on preserving structural information in multimodal embeddings [38]. The performance difference between the two autoregressive models highlights the impact of data balancing strategies. The model with balanced reduction (2,2) for audio and image data showed slightly better results, indicating potential overfitting with animated images. We now aim for a more versatile solution that can handle all aspects of data preparation and support true autoregressive modeling. Our `TokenPxByDataset` class remains valuable, but we are working to integrate its functionality more closely with our revised tokenizer for a more streamlined data pipeline.

While we explored both RNN and State Space Models (SSM), with SSMs showing promising rapid convergence [8], we're now focusing on simplifying the overall architecture. Our approach offers an alternative to models like ImageBind [11], as our results suggest it may be possible to unify modalities without relying on intermediate representations. We now see potential benefits in allowing emergent properties to develop within a simplified framework. Moving forward, we plan to refine our strategy to better use specific input positions for high-definition image and sound by including basic diffusion. Our exploration of diffusion-based models for setpoint control tasks has shown interesting results in terms of generalization, expanding the potential applications of our multimodal approach to control problems. This work also opens new possibilities for applying approaches like Diffusion-LM [22] and Autoregressive Diffusion Models [15], which may offer new perspectives on multimodal sequence modeling. While our work is ongoing, it aligns with recent trends in multimodal AI research [2] and could potentially provide a versatile foundation for various multimodal tasks.

References

[1] Michael Athans. On the determination of optimal costly measurement strategies for linear stochastic systems. *Automatica*, 8(4):397–412, 1972.

[2] 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.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[5] Nicolò Botteghi, Khaled Alaa, Mannes Poel, Beril Sirmacek, Christoph Brune, Abeje Mersha, and Stefano Stramigioli. Low dimensional state representation learning with robotics priors in continuous action spaces. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 190–197. IEEE, 2021.

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

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

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

[9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[10] Fabien Furfaro. Pixelbytes: A unified multimodal representation learning project. GitHub: `https://github.com/fabienfrfr/PixelBytes`, Models: `https://huggingface.co/ffurfaro/PixelBytes-Pokemon`, `https://huggingface.co/ffurfaro/aPixelBytes-Pokemon` and `https://huggingface.co/ffurfaro/aPixelBytes-OptimalControl`, Datasets: `https://huggingface.co/datasets/ffurfaro/PixelBytes-Pokemon`, `https://huggingface.co/datasets/ffurfaro/PixelBytes-PokemonAll` and `https://huggingface.co/datasets/ffurfaro/PixelBytes-OptimalControl`, 2024. GitHub repository, Hugging Face Model Hub, and Datasets Hub.

[11] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15180–15190, 2023.

[12] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

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

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[15] Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021.

[16] 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.

[17] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[18] Rudolf Emil Kalman et al. Contributions to the theory of optimal control. *Bol. soc. mat. mexicana*, 5(2):102–119, 1960.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[20] T Kudo. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[22] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022.

[23] Aobo Liang, Xingguo Jiang, Yan Sun, and Chang Lu. Bi-mamba4ts: Bidirectional mamba for time series forecasting. *arXiv preprint arXiv:2404.15772*, 2024.

[24] Victor Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Y Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. *Advances in Neural Information Processing Systems*, 35:17612–17625, 2022.

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

[26] Katsuhiko Ogata et al. *Modern control engineering*. Prentice Hall India, 2009.

[27] 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.

[28] 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.

[29] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

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

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

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

[33] Boubker Tabiti, Manon Leblanc, Joelle Ferzly, and Fabien Furfaro. Gymsetpoint: A library for reinforcement setpoint learning with lti systems. `https://github.com/fabienfrfr/Gym-Setpoint`, 2024.

[34] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12, 2016.

[35] 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.

[36] 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.

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

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

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

[40] Zhihao Wang, Jian Chen, and Steven CH Hoi. Deep learning for image super-resolution: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3365–3387, 2020.

[41] Fangneng Zhan, Yingchen Yu, Rongliang Wu, Jiahui Zhang, Kaiwen Cui, Changgong Zhang, and Shijian Lu. Auto-regressive image synthesis with integrated quantization. In *European Conference on Computer Vision*, pages 110–127. Springer, 2022.

[42] 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.

[43] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.