Fast Autoregressive Video Generation with Diagonal Decoding

Supplementary Material

1. Direct Comparison with ZipAR

We provide a straightforward implementation of ZipAR as a baseline on Cosmos, which means we only leverage spatial-level acceleration. Our experimental setup is consistent with the results shown in Table 1. The experimental results are as follows (ws stands for window size, which is a hyperparameter in ZipAR):

Two interesting findings further distinguish the superiority of Diagonal Decoding in video acceleration compared to the standard ZipAR approach.

- In the 4B model, by balancing spatial and temporal redundancy, DiagD are able to configure parameters such that the number of forward steps is much less than the ZipAR window size of 1 (which corresponds to the maximum acceleration ratio in ZipAR), while still achieving much better generation quality. In contrast, the ZipAR implementation shows severe image degradation under these conditions.
- In the 12B model, by fully exploiting temporal redundancy, we achieve a higher acceleration ratio than ZipAR(1.71 fps v.s. 1.04 fps) at comparable quality levels.

Our motivation incorporates both temporal(inter-frame) and spatial(intra-frame) redundancy (while ZipAR focuses only on spatial redundancy). For video tasks, temporal redundancy is clearly very important. By adjusting two hyperparameters, we can effectively balance spatial and temporal redundancy, achieving higher acceleration and better performance. Our analysis of the model's attention maps also supports this distinction, revealing that temporal redundancy plays a crucial role in accelerating video generation.

2. Detailed Explanation of Algorithm

In this section, we clarify our algorithms with examples and more detailed explanation.

2.1. Diagonal Decoding Algorithm

In the next token prediction algorithm, the Transformer receives one input token and produces the probabilities for the next token ($[t_1]$ß$[t_1, t_2]$ß$[t_1, t_2, t_3]$, ... However, this is just a conventional way of thinking; the Transformer is not limited to receiving only one token at a time. Consider the training phase of the Transformer, where it is actually fed a sequence of n tokens and learns to predict the probability of the next token for each token in the sequence. Therefore, moving from single-token generation to multi-token generation is not inherently difficult—in fact, this transition directly addresses the generation problem illustrated in the figure below. This explains how diagonal generation is performed.

$$\begin{bmatrix} t_1 & t_2 & \cdot \\ t_3 & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \Longrightarrow \begin{bmatrix} t_1 & t_2 & t_3 \\ t_4 & t_5 & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$$

Next, we discuss how to generate tokens correctly. The content of the tokens to be generated is actually controlled by the position embedding. In a Transformer model, each input token is assigned a position embedding to indicate its position in the sequence. The token to be generated corresponds to the next position for each input token. Therefore, by adding different position embeddings to the input token, we can predict the next token at different positions. For example, to predict t_5 , we simply use t_4 along with the position embedding corresponding to t_5.

Finally, we need to address how to generate the first token of each row, since the last token of the previous row has not yet been generated. By leveraging the spatiotemporal redundancy described in the paper, we select the preceding token as the input. Specifically, to predict t_7, we use token t_4 and the position embedding corresponding to t_6.

2.2. Fine-tuning Algorithm

The purpose of the attention mask is to address the inconsistency between visible tokens during training and inference. For example, in the case of t_7 shown above, causal attention masks in autoregressive training allow t_7 to attend to all tokens from t_1 to t_6. However, in diagonal decoding during inference, t_6 has not yet been generated when t_7 is produced. This creates some inconsistency, so the newly proposed attention mask ensures that t_7 cannot see t_6 during training either. Therefore, we only need to modify the traditional causal attention mask during Transformer training to achieve this, and finetuning can be performed in parallel. It is important to emphasize that, unless otherwise stated in the paper, all test results were obtained without finetuning.

3. Cosmos

3.1. Results on Robotic Manipulation and Driving

Open-ended scenarios indeed involve more complex motion dynamics, but there are few open-sourced autoregressive models performs well on open-ended data. We test DiagD on two more challenging datasets in Cosmos to demonstrate the generalization ability of Diagonal Decoding to open-ended

Table 1. Comparison between ZipAR and DiagD.

Model	Algorithm	FVD↓	Subject Cons.↑	Image. Qual.↑	Dynamic degree ↑	FPS↑	STEP (k)↓
4B	NTP	136	**0.978**	**0.604**	0.49	0.38	7.68
	ZipAR $ws = 1$	349	0.823	0.379	0.53	3.42	0.30
	DiagD $k = 2$ $d = 10$	137	**0.978**	0.599	0.46	**3.41**	0.15
12B	NTP	**135**	**0.978**	0.602	**0.54**	0.15	7.68
	ZipAR $ws = 1$	137	0.974	0.569	0.49	1.04	0.30
	DiagD $k = 1$ $d = 1$	139	0.967	0.564	0.51	1.71	**0.11**

scenarios: the Bridges dataset, which focuses on robotic manipulation, and the BDD100K dataset, which encompasses diverse driving environments. Both datasets feature highly significant, unpredictable, and pronounced motion as well as complex environments. We randomly selected 100 videos from the test set and randomly extracted 33 continuous frames from each video (no downsampling or frame interpolation was performed). We obtained the following results, Table 2 and 3. It is worth noting that the Cosmos model we used has not been post-trained on robotic or autonomous driving datasets, which limited the capabilities of both next-token prediction and Diagonal Decoding.

3.2. Details of Cosmos Models

Cosmos, a World Foundation Model (WFM) Platform for developing Physical AI systems, integrates multiple pre-trained models, including autoregressive and diffusion-based methods, as well as discrete and continuous tokenizers. Specifically, the autoregressive model employs a discrete video tokenizer that leverages a codebook containing $16,000$ entries, achieving spatial compression of $16\times$ and temporal compression of $8\times$. This tokenizer is capable of compressing a video of 33 frames at a resolution of 640×1024 into $12,800$ discrete tokens. In our study, We implements the spatial and temporal diagonal decoding algorithm on Cosmos autoregressive-based world foundation models. We provided an initial sequence of $5,120$ tokens (equivalent to 9 frames), optionally accompanied by text depending on the task, to evaluate text-guided video generation and video continuation tasks. This initial sequence was used to generate the subsequent $7,680$ tokens, extending the remaining frames to reach the 33-frame length.

Due to the lack of an open-source evaluation pipeline and datasets, we replicated a comparable setup based on details provided in its technical report. Specifically, we selected 100 videos, each comprising 33 frames, randomly sampled from the RealEstate10K dataset. To quantitatively assess visual quality, we employed standard metrics, including Fréchet Video Distance (FVD) and Subject Consistency, Dynamic Degree, and Image Quality from VBench. Furthermore, we conducted a human evaluation, detailed in Sec:analysis, to compare visual quality and object movement between videos generated by next-token prediction and DiagD. Unlike Cosmos, our evaluation metrics exclude the use of diffusion decoder for post-processing videos generated by autoregressive models, as this would not fairly reflect the visual quality.

3.3. Extra Experiments on Hyper-parameters

We report more combination of k and d in Table 4. The FPS values show minimal variation across some settings. This is because the implementation of the diagonal decoding algorithm introduces a small amount of overhead. When the speedup ratio is significantly high, the time lost due to this overhead becomes non-negligible.

3.4. More Cases

We randomly choose ten cases from 100 evaluation sets in supplementary material.

4. WHAM

4.1. Details of WHAM

The World and Human Action Model (WHAM) is a recently proposed state-of-the-art autoregressive generative model trained on gameplay data from *Bleeding Edge*, capable of generating coherent and diverse gameplay sequences based on user instructions. Unlike Cosmos, WHAM employs an image-level Vector Quantized (VQ) tokenizer that concentrates exclusively on spatial compression. This tokenizer independently converts each game state, with a resolution of 180×300, into 540 discrete tokens, which are subsequently concatenated with their corresponding in-game actions. To preserve the inherent relationship between actions and game states, our approach employs spatial diagonal decoding algorithm alone instead of processing the entire video simultane-

Table 2. Quantitative evaluation of Cosmos on BridgeV2. "NTP" refers to the next-token prediction paradigm. DiagD $k = i\ d = j$ denotes the Diagonal Decoding algorithm with different hyper-parameters. "STEP" refers to the number of forward passes required by the model to generate a video.

Model	Algorithm	FVD↓	Subject Cons.↑	Image. Qual.↑	Dynamic degree	FPS↑	STEP (k)↓
4B	NTP	602	0.926	0.594	0.86	0.38	7.68
	DiagD $k = 4\ d = 20$	615	0.928	0.592	0.87	2.66	0.26
	DiagD $k = 2\ d = 40$	625	0.921	0.585	0.65	3.02	0.30
	DiagD $k = 2\ d = 10$	626	0.921	0.580	0.69	**3.41**	0.15
12B	NTP	**585**	0.926	0.590	0.80	0.15	7.68
	DiagD $k = 2\ d = 40$	599	**0.930**	0.597	0.70	1.21	0.30
	DiagD $k = 2\ d = 10$	608	**0.930**	**0.598**	0.76	1.50	0.19
	DiagD $k = 1\ d = 1$	609	0.923	0.590	0.69	1.71	**0.11**

Table 3. Quantitative evaluation of Cosmos on BDD100k. "NTP" refers to the next-token prediction paradigm. DiagD $k = i\ d = j$ denotes the Diagonal Decoding algorithm with different hyper-parameters. "STEP" refers to the number of forward passes required by the model to generate a video.

Model	Algorithm	FVD↓	Subject Cons.↑	Image. Qual.↑	Dynamic degree	FPS↑	STEP (k)↓
4B	NTP	**567**	0.946	0.500	0.99	0.38	7.68
	DiagD $k = 4\ d = 20$	575	**0.947**	**0.501**	0.98	2.66	0.26
	DiagD $k = 2\ d = 40$	579	0.945	0.498	0.97	3.02	0.30
	DiagD $k = 2\ d = 10$	585	0.940	0.492	0.99	**3.41**	0.15
12B	NTP	569	0.946	**0.501**	0.99	0.15	7.68
	DiagD $k = 2\ d = 40$	568	**0.947**	0.499	0.99	1.21	0.30
	DiagD $k = 2\ d = 10$	568	0.943	0.494	0.99	1.50	0.19
	DiagD $k = 1\ d = 1$	581	0.926	0.462	0.99	1.71	**0.11**

ously. That is to say, we sequentially generate subsequent game states from previous states and their associated actions, alternating between state generation and action concatenation.

For WHAM, we randomly selected 100 videos from its evaluation set to assess video consistency according to WHAM's evaluation protocol. The generation of each video was conditioned on one second of gameplay, which included both video and controller actions, and then proceeded to be conditioned on the controller actions performed by a human player during the subsequent 10 seconds of gameplay. In addition to reporting Fréchet Video Distance (FVD) and Subject Consistency, Dynamic Degree, and Image Quality from VBench. Following WHAM's protocol, we conducted a human evaluation to assess the visual quality of generated gameplay and object motion, comparing our results with those obtained using the next-token prediction algorithm.

4.2. More Cases

We randomly choose ten cases of 10 senconds videos from 100 evaluation sets in supplementary material.

5. MC-AR

5.1. Details of MC-AR Models

We conducted a series of experiments by training models from scratch on the VPT dataset. The VPT dataset is a

Table 4. Quantitative evaluation on Cosmos. 4B and 12B refer to models used for video continuation. "NTP" refers to the next-token prediction paradigm. $DiagD\ d = m\ k = n$ denotes the Diagonal Decoding algorithm where $d = m$ and $k = n$. "Step" refers to the number of forward passes required by the model to generate a video. "TP" is the number of tokens that model can generate per second.

Model	Algorithm	FVD↓	Subject Cons.↑	Image. Qual.↑	Dynamic degree ↑	FPS↑	TP↑	STEP (k)↓
4B	$d = 2, k = 1$	348	0.844	0.393	0.52	**3.42**	**1280**	**0.11**
	$d = 3, k = 1$	350	0.846	0.397	0.54	3.41	1097	0.11
	$d = 5, k = 1$	352	0.859	0.402	0.51	3.41	1097	0.11
	$d = 9, k = 1$	342	0.863	0.408	0.52	3.41	1097	0.12
	$d = 4, k = 2$	171	0.936	0.515	**0.55**	3.41	1097	0.15
	$d = 6, k = 2$	145	0.966	0.574	0.49	3.41	1097	0.15
	$d = 10, k = 2$	137	0.978	0.599	0.46	3.41	1097	0.16
	$d = 18, k = 2$	**136**	**0.979**	0.601	0.48	3.00	960	0.24
	$d = 8, k = 4$	154	0.959	0.552	0.54	2.67	853	0.24
	$d = 12, k = 4$	139	0.976	0.595	0.49	2.67	853	0.24
	$d = 20, k = 4$	**136**	**0.979**	**0.604**	0.50	2.66	852	0.26
	$d = 36, k = 4$	**136**	**0.979**	0.603	0.48	2.40	768	0.29
12B	$d = 2, k = 1$	139	0.967	0.564	0.51	**1.71**	**549**	**0.11**
	$d = 3, k = 1$	152	0.955	0.546	0.53	1.71	549	0.11
	$d = 5, k = 1$	143	0.970	0.576	0.49	1.71	549	0.11
	$d = 9, k = 1$	**136**	0.863	0.408	**0.52**	1.61	515	0.12
	$d = 4, k = 2$	152	0.968	0.563	**0.52**	1.60	512	0.15
	$d = 6, k = 2$	143	0.973	0.585	0.50	1.60	512	0.15
	$d = 10, k = 2$	143	0.978	0.600	0.51	1.60	512	0.16
	$d = 18, k = 2$	**136**	**0.979**	0.601	0.48	1.50	480	0.18
	$d = 8, k = 4$	150	0.970	0.570	0.59	1.26	427	0.24
	$d = 12, k = 4$	140	0.975	0.594	0.50	1.26	404	0.24
	$d = 20, k = 4$	137	0.978	0.600	0.49	1.20	384	0.26
	$d = 36, k = 4$	**136**	**0.979**	**0.603**	0.48	1.09	349	0.29

domain-specific dataset comprising gameplay videos from *Minecraft*. We employed a pre-trained image VQ-VAE, an image-level tokenizer with a codebook containing $8,192$ entries, achieving a spatial compression ratio of $16\times$. To enhance visual quality, we subsequently fine-tuned the VQ-VAE on the VPT dataset. Our Transformer model was based on the LLaMA architecture and augmented with 3D Rotary Embeddings. We combine each game state tokens with the corresponding actions just like WHAM, so for each pair of game state and actions in the original input (x_i, a_i), the tokenizers will transfer them into a flat sequence of discrete ids as:

$$(t_{i*c+1}, \cdots, t_{(i+1)*c}, t_1^{a_i}, \cdots, t_n^{a_i}). \tag{1}$$

and c is the number of ids to represent each state, n is the number of actions. We trained our model on next token prediction tasks, enabling the model to predict future states based on previous game states and current action. We use the Adam optimizer with a cosine decay learning rate scheduler to train the model. Additionally, fine-tuning was performed for an extra $1,000$ steps on the same dataset.

For MC-AR, we selected 100 video clips from an unused subset of the evaluation set, each containing 16 frames, with the last 15 frames corresponding to actions in the gameplay. Each model generated 15 subsequent frames conditioned on the first frame of each clip and the 15 actions. These generated frames were then compared against the ground truth using the FVD, Subject Consistency, Dynamic Degree, and Image Quality metrics.

5.2. Model Configurations

We train three different sizes of the model within the LLaMA architecture: 300M, 700M, and 1.2B. We tune the hidden dimension, intermediate dimension, and the number of layers to achieve different model sizes. The configuration of these models are listed in Table 5. The hyperparameters of the optimizer used to train the model are listed in Table 6.

5.3. Extra Experiments

We provide models of three scales (300M Table 7, and 1.2B Table 8) to present additional results on MC-AR.

Table 5. The configuration of different size of models.

	Hidden dim	MLP dim	Num. Heads	Num. Layers
300M	1024	4096	16	20
700M	2048	4096	32	20
1.2B	2048	8192	32	20

Table 6. Optimization hyperparameters.

Hyperparameter	Value
Learning rate scheduler	cosine
Learning rate	$3e^{-4}$
Warm up steps	10000
Weight decay	0.1
Optimizer	AdamW
AdamW betas	$(0.9, 0.95)$
Maximum Positions	5376

Table 7. Quantitative evaluation on 300M MC-AR. We use DiagD ($k = 2$) in experiment.

Algorithm	FVD↓	Subject Cons.↑	Image. Qual.↑	Dynamic degree ↑	FPS↑	STEP (k)↓
NTP	**223**	**0.869**	**0.676**	0.98	1.08	5.04
DiagD $k = 2$ w/o FT	246	0.854	0.650	**0.99**	**3.98**	**0.75**
DiagD $k = 2$ w/ FT	233	0.845	0.648	0.98	**3.98**	**0.75**

Table 8. Quantitative evaluation on 1.2B MC-AR. We use DiagD ($k = 2, 4$) in experiment.

Algorithm	FVD↓	Subject Cons.↑	Image. Qual.↑	Dynamic degree ↑	FPS↑	STEP (k)↓
NTP	**203**	**0.866**	**0.677**	0.97	0.89	5.04
DiagD $k = 4$ w/o FT	246	0.857	0.645	**0.98**	1.42	1.14
DiagD $k = 2$ w/o FT	246	0.841	0.606	0.97	**1.98**	**0.75**
DiagD $k = 2$ w/ FT	227	0.853	0.661	**0.98**	**1.98**	**0.75**

5.4. More Cases

We randomly choose ten cases from 100 evaluation sets in supplementary material.

6. Derivations

We derive Equation in Method Section here. First, assume $\min\{h, w\} = h$, we have:

$$\begin{aligned} r_{\text{spa}} &= \frac{h \cdot w}{(h-1) \cdot k + w} \\ &= \frac{h}{\frac{h}{w} \cdot k - \frac{k}{w} + 1} \\ &\approx \frac{h}{\frac{h}{w} \cdot k + 1}. \end{aligned} \tag{2}$$

Where we assume $\frac{k}{w} \approx 0$ which is applicable for most of our cases. And as a result, the approximation achieves if $h \approx w$.

Similarly, we have:

$$\begin{aligned} r_{\text{diag}} &= \frac{T \cdot h \cdot w}{(T-1) \cdot h + h + w - 1} \\ &= \frac{T \cdot h \cdot w}{T \cdot h + w - 1} \\ &\approx \frac{w}{1 + \frac{w}{T \cdot h}} \\ &\approx w \end{aligned} \tag{3}$$

Where the approximation stands when $T * h \gg w$, which is applicable for most of video generation cases.