This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

MixANT: Observation-dependent Memory Propagation for Stochastic Dense Action Anticipation

Syed Talal Wasim[1,2] Hamid Suleman[1,2] Olga Zatsarynna[1,2] Muzammal Naseer[3] Juergen Gall[1,2]
[1]University of Bonn [2]Lamarr Institute of ML and AI [3]Khalifa University

Abstract

*We present MixANT, a novel architecture for stochastic long-term dense anticipation of human activities. While recent State Space Models (SSMs) like Mamba have shown promise through input-dependent selectivity on three key parameters, the critical forget-gate (**A** matrix) controlling temporal memory remains static. We address this limitation by introducing a mixture of experts approach that dynamically selects contextually relevant **A** matrices based on input features, enhancing representational capacity without sacrificing computational efficiency. Extensive experiments on the 50Salads, Breakfast, and Assembly101 datasets demonstrate that MixANT consistently outperforms state-of-the-art methods across all evaluation settings. Our results highlight the importance of input-dependent forget-gate mechanisms for reliable prediction of human behavior in diverse real-world scenarios. The project page is available at* https://talalwasim.github.io/MixANT/.

1. Introduction

Human activity anticipation represents a fundamental challenge in computer vision, requiring models to predict future human actions from partial observations. This capability is crucial for applications ranging from assistive technologies to autonomous driving systems, where timely intervention depends on accurate forecasting of human intentions and behaviors. In particular, long-term dense action anticipation [1, 2, 11] presents a particularly challenging problem: generating multiple and continuous, frame-by-frame predictions of future activities, often extending several minutes ahead. In order to handle the uncertainty for this task, stochastic approaches have been proposed [8, 35, 36, 43] that predict multiple plausible future samples. While recent approaches [35, 36, 43] use a diffusion model to generate multiple predictions for the same observations, these approaches differ in the architecture that is used within the diffusion model, including gated temporal convolutions [35]

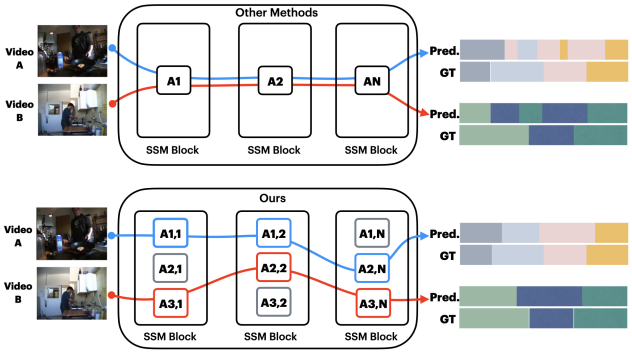


Figure 1. The current state-of-the-art method [36] employs standard Mamba blocks for stochastic dense action anticipation. In contrast, we introduce the MixMamba block, where each block dynamically selects an appropriate **A** matrix based on the input sequence.

and Transformers [43]. While Transformers have shown impressive capabilities in modeling long-range dependencies, their quadratic computational complexity with respect to sequence length limits their practical application to extended temporal horizons as they occur in long-term dense anticipation. MANTA [36] thus utilizes the Mamba architecture [12] due to its remarkable capabilities in handling long sequences with near-linear computational complexity.

The standard Mamba model [12] applies input-dependent selectivity to three of its four key parameters, dynamically adapting the model's behavior based on contextual cues within the observed sequence. Specifically, these parameters include the input gate, output gate, and time-step parameter. However, one critical parameter—the forget-gate (the **A** matrix) that controls how the hidden state evolves over time—lacks this input dependence. This forget-gate parameter determines how much past information is remembered or forgotten. While [36] demonstrated that input dependence on the input, output and time-step parameters is particularly important for long-term dense action anticipation, the fourth parameter, the matrix **A**, is independent of the input as illustrated in Fig. 1.

This raises a particularly important research question, as

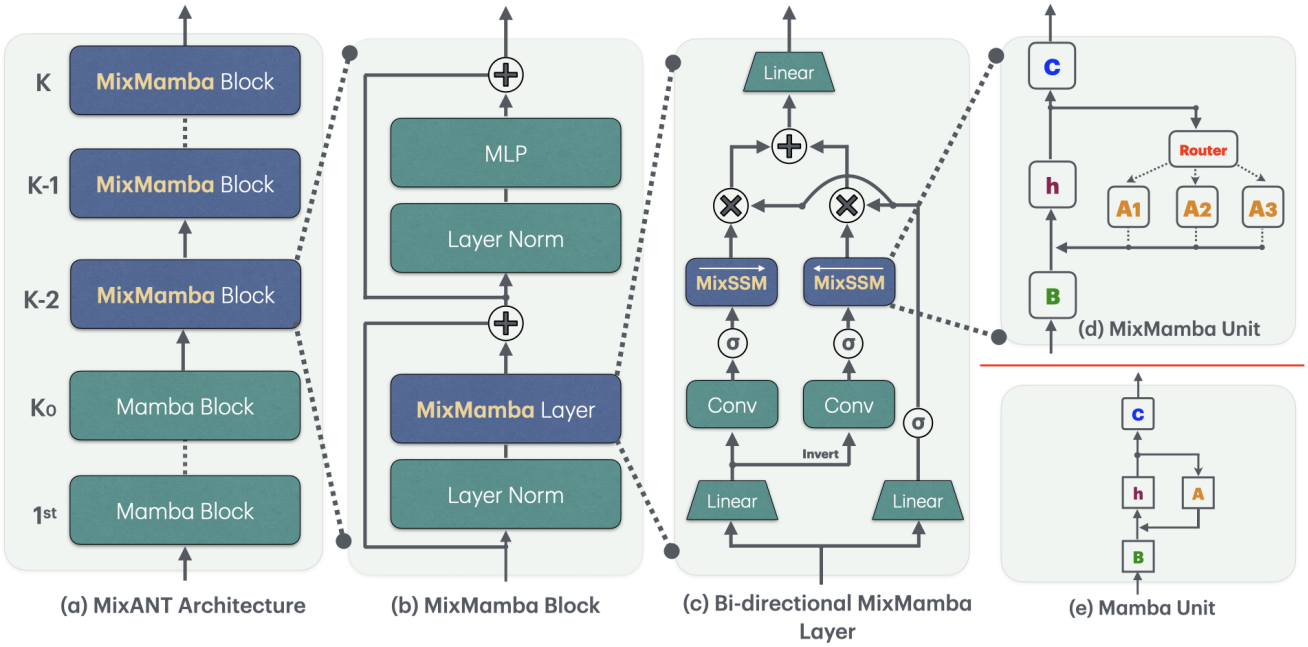


Figure 2. Our architecture and its components. In the MixANT architecture (a), the first K_0 blocks are standard Mamba blocks and the rest ($K - K_0$) are MixMamba blocks (b). Each MixMamba block contains a bi-directional MixMamba layer (c). In contrast to the standard Mamba unit (e) where $\mathbf{A}$ is fixed for all input sequences, the MixMamba unit (d) contains a router that selects the relevant $\mathbf{A}$ matrix depending on the input sequence.

the $\mathbf{A}$ matrix's role in controlling temporal memory is particularly relevant for anticipation tasks. Depending on the specific input context, sometimes past information is critically important, while in other contexts, it may be less relevant. Moreover, in the stochastic long-term dense anticipation task, the input sequence consists of observed data and padded zeros representing future unknown labels. In this scenario, it is also desirable to have selectivity on the hidden state through forget-gates such that it selectively ignores the padded zero sequence. Therefore, any form of input dependence on this matrix, that controls the evolution of the hidden state, is important. However, introducing input dependence to the $\mathbf{A}$ matrix presents substantial technical challenges. Direct computation would either require query-key like multiplication, which diminishes the sub-quadratic complexity advantage Mamba models, or a large MLP, which adds significant computational overhead.

In this work, we address this challenge by introducing MixANT, a mixture of $\mathbf{A}$ for ANTicipation, which enhances the network's representational capacity with minimal computational overhead. As shown in Fig. 2, our method employs layers called MixMamba that dynamically select the most contextually relevant $\mathbf{A}$ matrix based on input features. This approach introduces input dependence without sacrificing computational efficiency. To the best of our knowledge, we propose the first mixture of experts approach specifically for the $\mathbf{A}$ matrix in Mamba-based architectures.

We demonstrate that this novel architecture achieves state-of-the-art performance on stochastic long-term dense anticipation task through extensive experimentation on three benchmark datasets: 50Salads [32], Breakfast [18], and Assembly101 [29]. Our approach consistently outperforms existing methods across all evaluation settings, and our results highlight the potential of our approach to advance the field of stochastic dense action anticipation and enable reliable prediction of human behavior in diverse real-world scenarios.

2. Related Work

Action anticipation research [41] is divided into different branches, but mainly consists of short-term approaches (predicting actions within seconds) [9, 10, 24, 28, 33, 34, 38, 39, 42] and long-term approaches (forecasting multiple actions over minutes). Our work focuses on long-term dense anticipation, which requires predicting both action sequences and their durations across specified future frames. Unlike methods that treat anticipation as ordered [7, 23, 24, 38] or unordered [26, 38, 43] transcript prediction without temporal boundaries, our work focuses on long-term dense anticipation. This application requires pre-

dicting both action sequences and their durations across specified future frames.

Dense anticipation approaches further split into deterministic models [1, 11, 17, 28] that generate single predictions, and stochastic methods [8, 37] — including ours — that produce multiple plausible futures for each observation. Other methods, e.g. [35, 43], can generate predictions in both stochastic and deterministic settings. [8] pioneered stochastic dense anticipation by extending RNN architecture to generate multiple samples. [37] enhanced this framework using adversarial learning to improve prediction diversity. Recent approaches [35, 43] have advanced beyond requiring ground-truth actions by performing simultaneous past classification and future forecasting. Specifically, [43] combined a diffusion model with the FUTR encoder [11] to model uncertainty in future actions, while [35] introduced a gated temporal diffusion network that models uncertainty in both past and future actions, achieving state-of-the-art results.

Traditional attention-based architectures face efficiency challenges due to quadratic complexity with sequence length, inspiring the development of State Space Models [13, 30] that offer linear scaling with sequence length. Mamba [12] enhances this framework through selective state space modeling with input-dependent selection mechanisms, making it particularly effective for long sequence processing. This efficiency has led to Mamba's adaptation across various fields including anticipation [36], vision [19, 22], and as backbones for diffusion models in generative tasks [14, 25]. Recently, [36] also utilized bi-directional Mamba blocks similar to [19] for stochastic long-term anticipation. Our work uniquely leverages Mamba's capabilities by introducing a novel "mixture of **A**" matrices approach that addresses the shortcomings of [12] mentioned in the introduction.

The Mixture of Experts (MoE) architectures allow for significant scaling of model parameters with minimal computational overhead. First proposed by [15, 16], MoE leverages conditional computation, i.e., activating only specialized "expert" sub-networks for specific inputs. By implementing this sparse activation strategy, MoE-based models can substantially increase model parameters with reasonable computational demands. The MoE technique has been successfully applied to both transformer-based and, more recently, Mamba-based architectures. Similar to our architecture in spirit are Mamba-based methods leveraging a mixture of experts (MoE) [3, 20, 27]. However, these approaches differ from our method in key ways. Both [3] and [27] utilize standard Mamba blocks [12] while employing MoE only to MLPs outside the Mamba block. Meanwhile, [20] employs a mixture-based architecture for modality-aware selection but unlike our method it uses a shared **A** matrix (of Mamba block) constraining the expressivity of the network.

3. Observation-dependent Memory Propagation in Mamba

The work [36] showed that state-space models outperform transformer and other architectures for stochastic long-term dense action anticipation. The approach utilizes the bi-directional Mamba layer [12, 19] for modeling temporal relations between past observations and future actions. A key feature of this block compared to previous state-space models are the input-dependent parameters, as we will discuss in Sec. 3.1. The temporal memory propagation, however, which is steered by the matrix **A**, remains input-independent as it is illustrated in Fig. 1, *i.e.*, the learned matrices **A** for each block do not change for different observations. This poses a major limitation for action anticipation since the way how temporal memory should be propagated between observations and future actions depends on the context of the observation. Indeed, we will show that the matrices **A** vary depending on the semantic context, see Fig. 6.

We will first describe the Mamba layer [12, 19] in Sec. 3.1 and our extension in Sec. 3.2. The entire approach for stochastic long-term dense action anticipation is then described in Sec. 4.

3.1. Mamba Layer

State-space models define a mapping from an input sequence x_t to an output sequence y_t through a latent state h_t using a first-order ordinary differential equation discretized with a time-step parameter Δ:

$$h_t = \bar{\mathbf{A}} h_{t-1} + \bar{\mathbf{B}} x_t, \tag{1}$$

$$y_t = \mathbf{C} h_t, \tag{2}$$

$$\bar{\mathbf{A}} = \exp(\Delta \mathbf{A}), \tag{3}$$

$$\bar{\mathbf{B}} = (\Delta \mathbf{A})^{-1}(\exp(\Delta \mathbf{A}) - I)\Delta \mathbf{B}. \tag{4}$$

Mamba [12] enhances this framework by making most parameters input-dependent, *i.e.*, $\mathbf{B}(x) \in \mathbb{R}^{B\times T\times N}$, $\mathbf{C}(x) \in \mathbb{R}^{B\times T\times N}$, and $\Delta(x) \in \mathbb{R}^{B\times T\times D}$ are computed as functions of the input $x \in \mathbb{R}^{B\times T\times D}$ where B is batch-size, T is sequence length, N is latent state dimension, and D is channel dimension. Δ controls the flow of information from the input x_t to the hidden state h_t. $\bar{\mathbf{B}}$ modulates the input. $\bar{\mathbf{A}}$ serves as a *forget-gate*, determining which information persists from previous states and how quickly past information decays, effectively controlling the model's memory duration. Finally, the **C** parameter filters state information for outputs, determining which aspects of the internal state are expressed in the output signal. While [12] made the hypothesis that it is sufficient to make Δ input-dependent and keep **A** input-independent, we demonstrate that this is not true

for the task of long-term dense action anticipation, where very long sequences are processed.

3.2. MixMamba Layer

Making the $\mathbf{A}$ matrix input-dependent presents substantial computational challenges. A straightforward implementation would either require query-key multiplication, which would negate Mamba's sub-quadratic complexity advantage for long sequences, or necessitate a very large MLP, which would introduce significant computational overhead and parameter inefficiency. For dense anticipation tasks specifically, where models must effectively reason over both observed and unobserved regions of input sequences, this limitation is particularly restrictive. An input-dependent $\mathbf{A}$ matrix would enable more nuanced control over how past information is selectively retained or forgotten based on the content of observed tokens.

To address these challenges while maintaining computational efficiency, we propose a mixture-of-experts approach for the $\mathbf{A}$ matrix. Rather than computing a single input-dependent $\mathbf{A}$ matrix directly, we maintain multiple expert $\mathbf{A}$ matrices and employ a softmax gating mechanism to select the appropriate combination based on input characteristics.

Building upon the S6 algorithm [12], we propose S6$^+$, which enhances the selection mechanism by implementing a mixture approach for the $\mathbf{A}$ matrix. Instead of using a single fixed $\mathbf{A}$ matrix, we define a set of E expert matrices $\{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_E\} \in \mathbb{R}^{E \times D \times N}$. For a given input sequence, we compute a gating vector $\gamma(x) \in \mathbb{R}^{B \times E}$ based on the projected mean of the input tokens γ:

$$\gamma(x) = \text{softmax}(W_g \cdot \text{mean}(x)), \tag{5}$$

where $W_g \in \mathbb{R}^{D \times E}$ is a learnable projection matrix. The matrix $\mathbf{A}$ is then selected by:

$$\mathbf{A}(x) = \mathbf{A}_{\hat{e}}, \quad \hat{e} = \arg\max_e \gamma_e(x), \tag{6}$$

where $\gamma_e(x)$ is the e-th element of the routing vector $\gamma(x)$. Note that during training, both $\{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_E\}$ and $\gamma(x)$ are learned, but it requires an additional loss term as we will discuss in Sec. 4.3.

The general structure of our MixMamba layer follows the bi-directional approach [19] with the S6$^+$ algorithm as illustrated in Fig. 2(c). The processing flow for both paths is given by

$$\hat{F}_{\text{Fwd}} = \sigma(H^K * \text{FF}(F_t^l)), \quad \hat{F}_{\text{Bwd}} = \sigma(H^K * \text{FF}(\overleftarrow{F_t^l})), \tag{7}$$

where H^K is a 1D convolutional filter with size K, flipping across the temporal dimension is denoted as $\leftarrow$, and $*$ represents the convolution operator. For both forward and backward paths, we apply our mixture-of-experts approach:

$$W_t^l = \text{S6}^+(\hat{F}_{\text{Fwd}}), \quad B_t^l = \text{S6}^+(\hat{F}_{\text{Bwd}}). \tag{8}$$

Similar to the Mamba layer, we apply a residual gating mechanism and combine the outputs:

$$W_t^l = W_t^l \odot R_t^l, \quad B_t^l = \overleftarrow{B_t^l} \odot R_t^l, \tag{9}$$

$$O_t^l = \text{FF}(W_t^l + B_t^l) \tag{10}$$

where $R_t^l = \sigma(\text{FF}(F_t^l))$ and $\odot$ represents element-wise multiplication.

This mixture-of-experts approach for the $\mathbf{A}$ matrix enhances the model's ability to selectively process temporal information, which is crucial for tasks involving long sequences with varying temporal dependencies. By dynamically weighting different expert $\mathbf{A}$ matrices based on input characteristics, the S6$^+$ algorithm can adapt its temporal processing strategy to the specific patterns present in the data, enabling more effective modeling of complex sequential relationships.

4. MixANT Architecture for Stochastic Long-Term Dense Action Anticipation

In this section, we present our approach for long-term stochastic dense action anticipation that is based on the bi-directional MixMamba layer described above in Sec. 3.2. For a fair comparison, we follow the diffusion architecture [36], which we discuss in Sec. 4.1. In Sec. 4.2, we describe the proposed MixANT architecture, which is illustrated in Fig. 2(a). In Sec. 4.3, we finally describe the loss functions. Our final loss includes a novel load balancing loss, which encourages that the router in the MixMamba unit (Fig. 2(d)) utilizes and learns all matrices and not only a subset during training.

4.1. Diffusion for Stochastic Long-Term Dense Action Anticipation

Following [35], we employ a diffusion model framework for stochastic long-term dense action anticipation. The diffusion model learns the distribution of per-frame actions given a set of observed frames:

$$\tilde{Y}_{1:P+F} \sim p(Y_{1:P+F} \mid \mathbf{x}_{1:P}), \tag{11}$$

where P is the number of observed frames, F is the number of future frames to predict, $\mathbf{x}_{1:P}$ is the sequence of observed frames, $Y_{1:P+F}$ represents the random variables for each frame of the entire sequence, and $\tilde{Y}_{1:P+F}$ represents the sampled action label for each frame. To condition the anticipation on the observed visual frames, a conditioning vector $\mathcal{X}$ is created by extending the observed visual features with zeros for future frames:

$$\mathcal{X} = \{\phi(x_1), \ldots, \phi(x_P), \underbrace{0, \ldots, 0}_{F}\} \tag{12}$$

where $\phi(x_i)$ represents visual features extracted from the i-th observed frame. For a fair comparison, we use the same features that have been used in previous works. After sampling $\hat{Y}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the diffusion model gradually generates a sample $\hat{Y}_{1:P+F} = \hat{Y}_0$ after T iterations

$$\hat{Y}_{t-1} = G_\theta(\hat{Y}_t, \mathcal{X}, t), \tag{13}$$

based on an anticipation generator G_θ with learnable parameters θ. The index t denotes the diffusion step. While [35] uses gated temporal convolutions and [36] a Mamba architecture for the generator G_θ, we harness our proposed MixANT architecture, which we will describe in Sec. 4.2. During inference, multiple predictions are sampled using DDIM [31] for a single set of input frames $\mathbf{x}_{1:P}$. For more details regarding the diffusion model, we refer the reader to [35].

4.2. MixANT Architecture

Our proposed MixANT architecture consists of a sequence of state-space model blocks, where we utilize a hybrid approach combining standard Mamba blocks and our proposed MixMamba blocks. This design allows the model to extract basic features in the earlier layers and benefit from the enhanced selective capabilities in the later layers.

The overall architecture processes the input to predict future actions through a sequence of K total blocks. Among these, the first K_0 blocks utilize standard Mamba blocks, while the remaining $K_E = K - K_0$ blocks employ our proposed MixMamba blocks, as illustrated in Fig. 2(a). Each block follows a similar structure to transformer blocks, with normalization, state-space processing, and MLP components, as illustrated in Fig. 2(b).

Given pre-extracted observed frame features, we first create the conditioning vector $\mathcal{X}$ as described in Eq. (12). This vector is then concatenated with the latent variables $\hat{Y}_t$ along the channel dimension to form $F_t \in \mathbb{R}^{(P+F)\times(n_c+n_d)}$, where n_c is the number of classes and n_d is the number of feature channels. For the k^{th} block, the processing flow is as follows:

$$\hat{F}_t^k = \begin{cases} \text{Mamba}(\text{LN}(F_t^{k-1})), & \text{if } k \leq K_0 \\ \text{MixMamba}(\text{LN}(F_t^{k-1})), & \text{if } k > K_0, \end{cases} \tag{14}$$

where LN represents Layer Normalization [4]. It needs to be emphasized that $\gamma(F_{t,1:P}^{k-1})$ in Eq. (5) is only conditioned on the features of the observed frames. Ablation on conditioning MixMamba on both observed and future frames is provided in the suppl. material. After the state-space processing, an MLP layer is applied, followed by a residual connection:

$$F_t^k = \text{MLP}(\hat{F}_t^k) + F_t^{k-1} \tag{15}$$

After processing through all K blocks, we generate the final prediction $\hat{Y}_{t-1} \in \mathbb{R}^{(P+F)\times n_c}$ using an MLP layer, representing the per-frame action predictions for both observed and future frames.

4.3. Training

We train our MixANT model using the established diffusion training approach [35] with an additional load balancing mechanism to ensure effective utilization of all expert matrices $\{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_E\}$ in our mixture approach.

For each training step, we sample a diffusion step t uniformly at random and obtain the corresponding noisy action sequence $\tilde{Y}_t$ using the forward diffusion process. We then pass $\tilde{Y}_t$ alongside the conditioning vector $\mathcal{X}$ to our MixANT model to obtain a reconstruction $\hat{Y}_0$ of the ground-truth action sequence Y. The primary reconstruction loss is the $\mathcal{L}_2$-loss between the predicted and the one-hot-encoded ground-truth action sequences:

$$\mathcal{L}_{rec} = ||Y - \hat{Y}_0||^2. \tag{16}$$

To ensure balanced utilization of the expert matrices in our mixture approach, we propose a load-balancing mechanism. During training, we track the usage of each expert by computing:

$$C_e^k = \sum_{b=1}^{B} \gamma_e^k(F_{t,1:P}^{k-1}(b)) \tag{17}$$

where C_e^k represents the weight for expert e in layer k in the batch of size B. The load balancing loss then encourages uniform expert utilization by minimizing:

$$\mathcal{L}_{lb} = \sum_{k=K_0+1}^{K} \text{KL}\left(\frac{C^k}{\sum_e C_e^k} \middle\| \mathcal{U}(E)\right) \tag{18}$$

where $\mathcal{U}(E)$ represents a uniform distribution over E experts, and KL denotes the Kullback-Leibler divergence.

Our combined training objective is then given by

$$\mathcal{L}_{total} = (1 - \lambda_{lb})\mathcal{L}_{rec} + \lambda_{lb} \cdot \mathcal{L}_{lb}, \tag{19}$$

where λ_{lb} controls the contribution of the load balancing loss. The impact of λ_{lb} is evaluated in the suppl. material.

5. Experiments

Evaluation Settings: We evaluate our proposed MixANT model using the stochastic long-term dense action anticipation protocol introduced by Farha *et al.* [8]. In this protocol, the observation length and anticipation horizon are specified as proportions of the total video duration. For a video v with n_v frames, the model observes the first

$P = \alpha n_v$ frames and predicts action labels for the subsequent $F = \beta n_v$ frames, where α and β represent the observation and anticipation ratios. For each observed video segment, we generate S=25 prediction samples from our model. We report two key metrics: Mean MoC (Mean over Classes) accuracy and Top-1 MoC accuracy. Mean MoC measures the average accuracy across all S generated samples, while Top-1 MoC reflects the accuracy of the best-matching sample. Consistent with previous works, we evaluate with observation ratios $\alpha \in \{0.2, 0.3\}$ and anticipation horizons $\beta \in \{0.1, 0.2, 0.3, 0.5\}$. Our experiments utilize three standard datasets for action anticipation:

Breakfast [18] comprises 1,712 cooking videos showing 10 different breakfast meal preparations across various kitchen environments. The videos are annotated with 48 action classes and vary in length, with the longest sequence requiring anticipation of up to 5.4 minutes. We evaluate using the established 4-fold cross-validation and report average performance across these splits.

Assembly101 [29] is a large-scale dataset containing 4,321 videos of toy assembly tasks from multiple viewpoints. The videos are densely annotated with 202 coarse action classes, with the longest sequences extending beyond 25 minutes and requiring anticipation up to 12.5 minutes into the future. Following standard protocol, we train on the training set and evaluate on the validation set.

50Salads [32] consists of 50 videos showing salad preparation, annotated with 17 action classes. The mean video duration is 6.4 minutes, with the longest sequence requiring anticipation of up to 5.1 minutes. We report results averaged over the 5 standard cross-validation splits.

For a fair comparison with previous methods, we use identical pre-extracted frame features. Specifically, we utilize I3D features from [2] and [11] for Breakfast and 50Salads datasets, while for Assembly101 we employ TSM-features [21] provided by [29].

Implementation Details: Our MixANT architecture consists of $K = 15$ sequential processing blocks. To balance computational efficiency with the benefits of our mixture approach, we implement a hybrid design where the first $K_0 = 3$ blocks use standard bidirectional Mamba operations without mixtures, while the remaining $K_E = 12$ blocks incorporate our proposed mixture mechanism with $E = 5$ experts. This configuration allows the model to establish foundational representations in the initial layers while leveraging the enhanced selective capabilities of the mixture approach in the deeper layers. Additional implementation details are provided in the suppl. material.

5.1. Comparison to State of the Art

Tab. 1 presents a comparison of our MixANT approach against previous methods across all three datasets. Our model consistently outperforms existing techniques, establishing new state-of-the-art results across virtually all evaluation settings and metrics.

MoC	Method	β ($\alpha = 0.2$)				β ($\alpha = 0.3$)			
		0.1	*0.2*	*0.3*	*0.5*	*0.1*	*0.2*	*0.3*	*0.5*
		Breakfast							
Mean	Tri-gram [8]	15.4	13.7	12.9	11.9	19.3	16.6	15.8	13.9
	UAAA [8]	15.7	14.0	13.3	13.0	19.1	17.2	17.4	15.0
	DiffAnt [43]	24.7	22.9	22.1	22.3	30.9	30.2	28.9	27.5
	GTDA [35]	24.0	22.0	21.4	20.6	29.1	26.8	25.3	24.2
	MANTA [36]	27.7	25.3	24.6	23.8	34.2	30.9	29.1	27.7
	MixANT	**29.6**	**26.3**	**25.9**	**25**	**36.2**	**32.8**	**31.2**	**28.7**
Top-1	Tri-gram [8]	-	-	-	-	-	-	-	-
	UAAA [8]	28.9	28.4	27.6	28.0	32.4	31.6	32.8	30.8
	DiffAnt [43]	31.3	29.8	29.4	30.1	37.4	37.0	36.3	34.8
	GTDA [35]	51.2	47.3	45.6	45.0	54.0	50.4	49.6	47.8
	MANTA [36]	55.5	51.0	47.9	46.9	59.6	55.0	53.7	51.9
	MixANT	**57.1**	**52.0**	**49.1**	**48.4**	**60.7**	**56.3**	**55.5**	**53.5**
		Assembly101							
Mean	Tri-gram [35]	2.8	2.2	1.9	1.5	3.5	2.7	2.3	1.8
	UAAA [8]	2.7	2.1	1.9	1.7	2.4	2.1	1.9	1.7
	GTDA [35]	6.4	4.5	3.5	2.8	5.9	4.2	3.5	2.9
	MANTA [36]	6.7	5.3	4.2	3.5	6.6	4.7	4.2	3.5
	MixANT	**8.0**	**7.0**	**6.2**	**4.4**	**8.7**	**6.6**	**5.6**	**4.6**
Top-1	Tri-gram [35]	9.0	8.0	7.2	5.6	9.5	8.2	7.8	5.9
	UAAA* [8]	6.9	5.9	5.6	5.1	5.9	5.5	5.2	4.9
	GTDA [35]	18.0	12.8	9.9	7.7	16.0	11.9	10.2	7.7
	MANTA [36]	16.9	13.3	10.2	8.8	15.6	12.0	11.1	8.4
	MixANT	**20.3**	**14.7**	**11.3**	**9.8**	**18.2**	**13.8**	**13.1**	**10.2**
		50Salads							
Mean	Tri-gram [8]	21.4	16.4	13.3	9.4	24.6	15.6	11.7	8.6
	UAAA [8]	23.6	19.5	18.0	12.8	28.0	18.0	14.8	12.1
	GTDA [35]	28.3	22.1	17.8	11.7	29.9	18.5	14.2	10.6
	MANTA [36]	28.6	22.8	19.5	13.6	31.3	21.9	17.6	13.0
	MixANT	**30.3**	**25.0**	**20.9**	**15.2**	**33.4**	**23.7**	**19.7**	**14.6**
Top-1	Tri-gram [8]	-	-	-	-	-	-	-	-
	UAAA* [8]	53.5	43.0	40.5	33.7	56.4	42.8	35.8	30.2
	GTDA [35]	69.6	55.8	45.2	28.1	66.2	44.9	39.2	31.0
	MANTA [36]	68.3	51.5	41.7	31.3	71.7	53.3	43.8	31.1
	MixANT	**71.5**	**56.9**	**46.5**	**35.0**	**72.9**	**54.6**	**44.9**	**32.4**

Table 1. Comparison of MixANT to state-of-the-art methods on Breakfast, Assembly101, and 50Salads.

On the Breakfast dataset, MixANT demonstrates substantial improvements over prior approaches, including MANTA [36] and GTDA [35]. For both Mean MoC and Top-1 MoC metrics, our method shows consistent gains across all anticipation horizons. Notably, for Mean MoC with α=0.3 and β=0.1, MixANT achieves 36.2%, compared to MANTA's 34.2%, while for Top-1 MoC, we achieve 60.7% versus 59.6%. The challenging Assembly101 dataset, with its 202 action classes, presents a more complex anticipation scenario where MixANT demonstrates even more substantial improvements. For Mean MoC, our approach shows relative gains of up to 32% over MANTA at longer anticipation horizons. For Top-1 MoC, MixANT surpasses the previous state-of-the-art GTDA in most settings, achieving 20.3% with α=0.2 and β=0.1.

On the 50Salads dataset, MixANT again establishes new state-of-the-art results. Our method achieves the highest performance for both Mean MoC across all settings and Top-1 MoC for nearly all configurations, with particularly

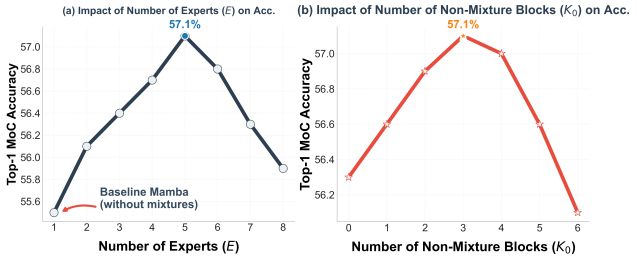


Figure 3. (a) Impact of number of experts in each layer. (b) Effect of the number of initial static blocks (K_0).

strong performance at shorter anticipation horizons. These consistent improvements across diverse datasets demonstrate the effectiveness of our mixture-based approach for the state-transition matrix. By enabling adaptive temporal memory processing through expert specialization, MixANT significantly enhances the model's capacity to generate accurate and contextually relevant action predictions across varying time scales.

5.2. Analysis

In this section, we discuss a series of analysis and ablation experiments that we performed to explore individual components of our proposed MixANT model. All experiments use the Breakfast dataset with α=0.2 and β=0.1.

Number of experts per block: A design choice in our work is the number of experts per block. MixANT allows for flexibility in this parameter, starting from a single expert (equivalent to a standard Mamba block) and scaling up as needed. We conducted a systematic experiment, increasing the number of experts from 1 to 8. As shown in Fig. 3(a), the model performance improves significantly as we increase the number of experts from the baseline (*i.e.* 1 expert), reaching optimal results at approximately 5 experts per block. Beyond this point, we observe a performance decline as additional experts are introduced. This pattern reveals important insights about expert-based architectures. The initial performance gains can be attributed to two mechanisms: (1) increased model capacity and expressiveness through parallel processing pathways, and (2) expert specialization across diverse input patterns. Each expert effectively learns to handle a distinct subset of the data distribution, collectively enabling more nuanced representations than a single expert. However, the performance degradation beyond 5 experts highlights a fundamental training challenge in mixture-of-experts architectures. With a fixed training budget, models with excessive experts suffer from sparse gradient updates. Each expert receives fewer training signals, creating an effective data starvation that impedes further gains in performance. This observation suggests that scaling expert count should be proportional to available training data and computational resources.

Number of initial static blocks: Another design parameter in our architecture is the number of initial static blocks. These blocks implement standard Mamba processing with a single expert, before transitioning to our proposed mixture-of-experts configuration. To determine the optimal setup, we systematically investigated the number of initial static blocks from 0 to 6, as shown in Fig. 3(b). Our results reveal a clear pattern: performance improves significantly as we increase the number of initial static blocks up to 3, after which we observe a consistent decline. This finding demonstrates that expert routing is most effective when applied to intermediate and higher-level representations rather than raw input features. The information processing hierarchy within the network can explain this behavior. Early layers typically extract general, low-level features that benefit from consistent processing across all inputs. Introducing input-dependent routing too early forces premature specialization before the model has extracted meaningful features to guide expert selection. The initial static blocks provide a foundation of general feature extraction, allowing subsequent expert layers to make more informed routing decisions based on these refined representations. Conversely, having too many static blocks (beyond 3) constrains the network's expressiveness by limiting the layers that can benefit from adaptive, input-dependent processing. Each expert block adds capacity and flexibility to the model, so reducing their number effectively caps the model's ability to specialize across diverse input patterns. These results also highlight the importance of making the **A** matrix input-dependent since not introducing enough **A** matrices or experts hinders the model's performance.

Router configuration: In the MixMamba layer (Fig. 2(c)), there are two MixSSM units. One processes the input in the forward direction and the other one in the backward direction. Each of these MixSSM units has its own mixture of **A** matrices. In this case, the routing can be done separately (independent case) or collectively for both units (unified case). In separate routing, there are two learnable projection matrices W_g^1 and W_g^2 from which two gating vectors γ^1 and γ^2 are computed. In this case, the **A** matrix chosen for the forward and backward units depends on the index of the maximum values of γ^1 and γ^2 after the softmax operation, respectively. In the unified case, we compute only one gating vector γ which is then used for both forward and backward units. Our results in Fig. 4(b), show that the unified router configuration performs better than the independent one. This is intuitive since in independent configuration, bi-directionality of the SSM is sacrificed because the forward and backward units are learning two different **A** matrices instead of learning the $\overrightarrow{A}$ matrix for the forward direction and its counterpart $\overleftarrow{A}$ for the backward direction.

Load balancing and expert usage: The load balancing mechanism explicitly encourages uniform utilization of all

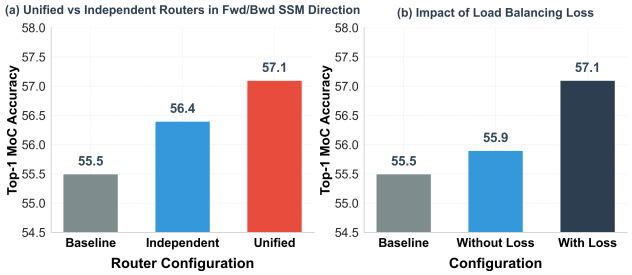


Figure 4. (a) Comparison of router configurations: unified (same experts in forward and backward passes) versus independent (different experts selected in forward and backward passes). (b) Impact of load balancing loss.

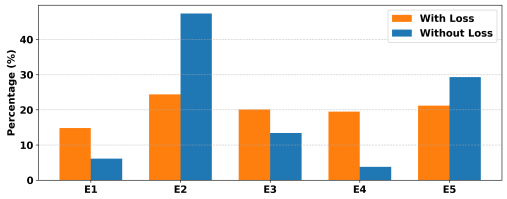


Figure 5. Utilization of the 5 experts across all layers on Breakfast.

experts during training as detailed in Sec. 4.3. This constraint is removed during inference. As demonstrated in Fig. 4(b), even without load balancing, our expert-based architecture outperforms the baseline due to its expanded representational capacity. However, these gains are significantly enhanced when load balancing is applied, as it ensures all experts receive adequate training signals to develop specialized functionality as shown in Fig. 5. Without load balancing, the second expert is selected in nearly 50% of the cases, while E1 and E4 are barely selected. The optimal utilization of the available expert capacity with load-balancing translates directly to improved performance metrics, validating the importance of balanced expert training in the MixANT architecture.

Additional ablations: In the suppl. material, we provide additional ablation experiments regarding the conditioning of the gating vector γ in (17), impact of λ_{lb} in (19), and a comparison of the proposed mixture-of-experts approach to a query-key approach or a very large MLP for making $\mathbf{A}$ input-dependent.

5.3. MixANT's Expert Selection Patterns

To understand MixANT's operational dynamics, we analyze expert selection patterns using the test split 1 of Breakfast with α=0.2 and β=0.1. For each testing instance, we initialize a selection matrix $S \in \mathbb{R}^{(K_E \times E)}$, where K_E represents the total number of mixture blocks and E is the number of experts per block. As input sequences propagate through the network, we track which expert $\mathbf{A}$ matrix is selected at each block. For each block row in S, we set a value of 1 in the column corresponding to the chosen expert while other columns remain zero. After processing each instance, we flatten these matrices and visualize them using t-SNE dimensionality reduction in Fig. 6.

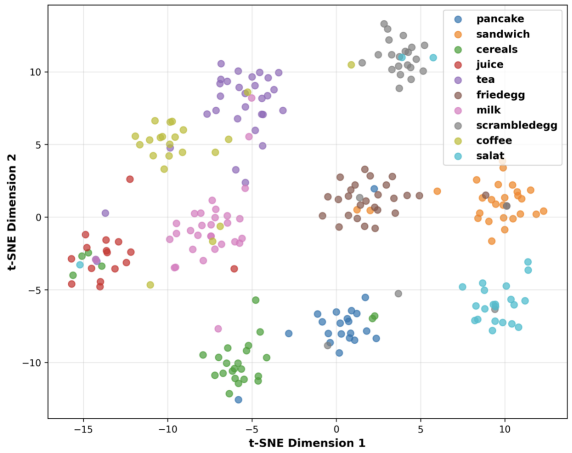


Figure 6. The t-SNE plot shows that using only a binary vector of the selected experts for each input sequence separates high-level activities. This shows that the selection of the experts in the MixANT model depends on the semantic context of the video.

The results reveal a striking emergent property: despite being trained only on atomic actions (cutting, stirring, pouring) rather than complete activity categories (making salad, preparing tea), MixANT implicitly learns to contextualize these actions within broader activities. The t-SNE visualization shows a clear clustering of expert selection patterns according to high-level activity categories, even without explicit training on these umbrella categories. This organization indicates that MixANT's expert matrices develop specialized functions aligned with semantic activity structures, enabling more nuanced temporal reasoning.

6. Conclusion

In this work, we presented MixANT, a novel architecture that enhances Mamba-based models for stochastic long-term dense action anticipation by introducing input-dependent selectivity to the forget-gate ($\mathbf{A}$ matrix). Our mixture of experts approach dynamically selects the most contextually relevant $\mathbf{A}$ matrix based on input features, providing crucial temporal memory control without sacrificing computational efficiency. Extensive experiments on the 50Salads, Breakfast, and Assembly101 datasets demonstrate that MixANT consistently outperforms existing state-of-the-art methods across all evaluation settings. These results confirm our hypothesis that input-dependent control of the hidden state evolution is particularly beneficial for anticipation tasks where the relevance of past information varies greatly across contexts.

Acknowledgement

The work has been supported by the ERC Consolidator Grant FORHUE (101044724), the Federal Ministry of Education and Research (BMBF) under grant no. 01IS22094A WEST-AI, the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) GA 1927/4-2 (FOR 2535 Anticipating Human Behavior), and the project iBehave (receiving funding from the programme "Netzwerke 2021", an initiative of the Ministry of Culture and Science of the State of North Rhine-Westphalia). For the computations involved in this research, we acknowledge EuroHPC Joint Undertaking for awarding us access to Leonardo at CINECA, Italy, through EuroHPC Regular Access Call - proposal No. EHPC-REG-2025R01-218.

References

[1] Y. Abu Farha, A. Richard, and J. Gall. When will you do what?-Anticipating temporal occurrences of activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 3

[2] Y. Abu Farha, Q. Ke, B. Schiele, and J. Gall. Long-term anticipation of activities with cycle consistency. In *DAGM German Conference on Pattern Recognition (GCPR)*, 2020. 1, 6, 2

[3] Quentin Anthony, Yury Tokpanov, Paolo Glorioso, and Beren Millidge. Blackmamba: Mixture of experts for state-space models. *arXiv preprint arXiv:2402.01771*, 2024. 3

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[5] Guo Chen, Yifei Huang, Jilan Xu, Baoqi Pei, Zhe Chen, Zhiqi Li, Jiahao Wang, Kunchang Li, Tong Lu, and Limin Wang. Video mamba suite: State space model as a versatile alternative for video understanding. *arXiv preprint, arXiv:2403.09626*, 2024. 4

[6] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision*, 2022. 4

[7] Srijan Das and Michael S Ryoo. Video+ clip baseline for ego4d long-term action anticipation. *arXiv preprint arXiv:2207.00579*, 2022. 2

[8] Y. Farha and J. Gall. Uncertainty-aware anticipation of activities. In *IEEE International Conference on Computer Vision Workshop (ICCVW)*, 2019. 1, 3, 5, 6

[9] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2020. 2

[10] R. Girdhar and K. Grauman. Anticipative Video Transformer. In *ICCV*, 2021. 2

[11] D. Gong, J. Lee, M. Kim, S.J. Ha, and M. Cho. Future transformer for long-term action anticipation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 3, 6, 2

[12] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 1, 3, 4

[13] Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. 3

[14] Vincent Tao Hu, Stefan Andreas Baumann, Ming Gui, Olga Grebenkova, Pingchuan Ma, Johannes Fischer, and Björn Ommer. Zigma: A dit-style zigzag mamba diffusion model. In *ECCV*, 2024. 3

[15] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 1991. 3

[16] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the em algorithm. In *International Joint Conference on Neural Networks (IJCNN)*, 1993. 3

[17] Q. Ke, M. Fritz, and B. Schiele. Time-conditioned action anticipation in one shot. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[18] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2, 6

[19] Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. Videomamba: State space model for efficient video understanding. In *European Conference on Computer Vision*, 2025. 3, 4

[20] Weixin Liang, Junhong Shen, Genghan Zhang, Ning Dong, Luke Zettlemoyer, and Lili Yu. Mixture-of-mamba: Enhancing multi-modal state-space models with modality-aware sparsity. *arXiv preprint, arXiv:2501.16295*, 2025. 3

[21] J. Lin, C. Gan, K. Wang, and S. Han. Tsm: Temporal shift module for efficient and scalable video understanding on edge devices. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 6, 1, 2

[22] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024. 3

[23] Esteve Valls Mascaró, Hyemin Ahn, and Dongheui Lee. Intention-conditioned long-term human egocentric action anticipation. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6048–6057, 2023. 2

[24] Himangi Mittal, Nakul Agarwal, Shao-Yuan Lo, and Kwonjoon Lee. Can't make an omelette without breaking some eggs: Plausible action anticipation using large video-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18580–18590, 2024. 2

[25] Shentong Mo and Yapeng Tian. Scaling diffusion mamba with bidirectional ssms for efficient image and video generation. *arXiv preprint arXiv:2405.15881*, 2024. 3

[26] Megha Nawhal, Akash Abdu Jyothi, and Greg Mori. Rethinking learning approaches for long-term action anticipation. In *European Conference on Computer Vision (ECCV)*, pages 558–576, 2022. 2

[27] Maciej Pióro, Kamil Ciebiera, Krystian Król, Jan Ludziejewski, Michał Krutul, Jakub Krajewski, Szymon Antoniak, Piotr Miłoś, Marek Cygan, and Sebastian Jaszczur. Moemamba: Efficient selective state space models with mixture of experts. *arXiv preprint arXiv:2401.04081*, 2024. 3

[28] Fadime Sener, Dipika Singhania, and Angela Yao. Temporal aggregate representations for long-range video understanding. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 3

[29] F. Sener, D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, and A. Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 6, 1

[30] Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations (ICLR)*, 2023. 3

[31] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 5

[32] Sebastian Stein and Stephen J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2013. 2, 6

[33] O. Zatsarynna and J. Gall. Action anticipation with goal consistency. In *IEEE International Conference on Image Processing (ICIP)*, 2023. 2

[34] Olga Zatsarynna, Yazan Abu Farha, and Juergen Gall. Multimodal temporal convolutional network for anticipating actions in egocentric videos. In *CVPR*, 2021. 2

[35] Olga Zatsarynna, Emad Bahrami, Yazan Abu Farha, Gianpiero Francesca, and Juergen Gall. Gated temporal diffusion for stochastic long-term dense anticipation. *European Conference on Computer Vision (ECCV)*, 2024. 1, 3, 4, 5, 6

[36] Olga Zatsarynna, Emad Bahrami, Yazan Abu Farha, Gianpiero Francesca, and Juergen Gall. Manta: Diffusion mamba for efficient and effective stochastic long-term dense anticipation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1, 3, 4, 5, 6

[37] H. Zhao and R. P. Wildes. On diverse asynchronous activity anticipation. In *European Conference on Computer Vision (ECCV)*, 2020. 3

[38] Qi Zhao, Shijie Wang, Ce Zhang, Changcheng Fu, Minh Quan Do, Nakul Agarwal, Kwonjoon Lee, and Chen Sun. Antgpt: Can large language models help long-term action anticipation from videos? *arXiv preprint arXiv:2307.16368*, 2023. 2

[39] Yue Zhao and Philipp Krähenbühl. Real-time online video detection with temporal smoothing transformers. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[40] Yue Zhao and Philipp Krähenbühl. Real-time online video detection with temporal smoothing transformers. In *ECCV*, 2022. 4

[41] Zeyun Zhong, Manuel Martin, Michael Voit, Juergen Gall, and Juergen Beyerer. A survey on deep learning techniques for action anticipation. *ArXiv*, 2023. 2

[42] Zeyun Zhong, David Schneider, Michael Voit, Rainer Stiefelhagen, and Jürgen Beyerer. Anticipative feature fusion transformer for multi-modal action anticipation. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023. 2

[43] Zeyun Zhong, Chengzhi Wu, Manuel Martin, Michael Voit, Juergen Gall, and Jürgen Beyerer. Diffant: Diffusion models for action anticipation. *ArXiv*, 2023. 1, 2, 3, 6