# Supplementary Material: Uncertainty-aware Action Decoupling Transformer for Action Anticipation

Hongji Guo[1*] Nakul Agarwal[2] Shao-Yuan Lo[2] Kwonjoon Lee[2] Qiang Ji[1]
[1]Rensselaer Polytechnic Institute, [2]Honda Research Institute, USA
{guoh11, jiq}@rpi.edu, {nakul_agarwal, shao-yuan_lo, kwonjoon_lee}@honda-ri.com

This document contains the supplementary materials of Uncertainty-aware Action Decoupling Transformer (UADT). We first include the implementation details (§ 1). Then we make a comparison of computational efficiency (§ 2). Next, we show the ablation study of decoder loss (§ 3), followed by the qualitative results (§ 4). Finally, we summarize the full inference workflow of UADT (§ 5).

## 1. Implementation Details

In this section, we provide the implementation details of UADT including feature extraction (§ 1.1), model architecture (§ 1.2), and fusion strategies (§ 1.3).

### 1.1. Feature extraction

#### 1.1.1 EPIC-KITCHENS 100

**RGB features.** We first perform a pre-processing of raw frames. The frames are center cropped of $224 \times 224$. Then we applied a normalization with mean $[0.45, 0.45, 0.45]$ and standard deviation $[0.25, 0.25, 0.25]$. No data augmentations are applied. We used the MViT-b [4] as the backbone for feature extraction. Specifically, we used Kinetics-400 [1] and Kinetics-700 [2] to pretrain the backbone for action classification. For Kinetics-400, we set the "frame length $\times$ sample rate" as $16 \times 4$. The frames are extracted at 30 FPS. 16 frames selected 4 frames apart are used to train the backbone. This leads to 2 seconds for each clip at 8 FPS. For Kinetics-700, we set the "frame length $\times$ sample rate" as $32 \times 3$. The frames are also extracted at 30 FPS. 32 frames selected 3 frames apart are used to train the backbone.

**Optical flow and object features.** In the main paper, we also include additional modalities to further boost the performance of UADT. Followed the procedures in [6], the optical flows are extracted by TVL1 algorithm [16]. Then a pretrained Batch Normalized Inception (BNInception) CNN is used to extract features. For the object-based features, a Faster R-CNN [9] pretrained on EPIC-KITCHENS-55 [3] is used to detect the active object at each frame. The class-based score is used as the features. We used the results from RU-LSTM (https://github.com/fpv-iplab/rulstm). When performing the multi-modality fusion, we simply concatenate feature vectors of different modalities at each time step.

#### 1.1.2 EGTEA Gaze+ and 50-Salads

**EGTEA Gaze+.** A pretrained TSN [14] is used to extract features. The TSN is pretrained on ImageNet-1k. Frames are extracted at 24 FPS. We followed the procedures in [6] (https://github.com/fpv-iplab/rulstm).

**50-Salads.** Following prior work [5, 12], we used the I3D [1] for feature extraction. Frames are extracted at 30 FPS and downsampled to 15 FPS. 3D convolution is used to capture the spatio-temporal dependencies among different frames.

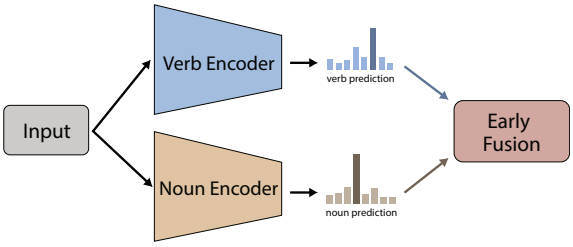
### 1.2. Model architecture

**Encoders.** The encoders of UADT are based on the transformer encoder [13]. Given the feature vectors $\boldsymbol{F}_t = \{\boldsymbol{f}_1, ..., \boldsymbol{f}_t\}$, a positional encoding is first added onto $\boldsymbol{F}_t$ to keep the original ordinal information. Each input token is linearly projected to queries, keys, and values. Then we perform multi-head attention among the inputs. Specifically, we set the number of heads as 4. After the multi-head attention, the intermediate embeddings are batch normalized and go through the feed forward network. Different from deterministic transformer, the feed forward networks in UADT encoders are probabilistic. Specifically, we assume the parameters follow Gaussian distributions. This is because Gaussian distribution has strong modeling capability for different distributions and we can easily train the model through backpropagation using reparameterization trick [11]. When performing the forward process, we can simply sample the parameter $\theta$ as $\theta = \mu + \epsilon\sigma, \epsilon \sim \mathcal{N}(0, 1)$, where $\mu$ is the learned mean, $\sigma$ is the learned standard deviation, and $\epsilon$ is a random variable that follows the unit normal distribution. In this way, the model can generate dif-

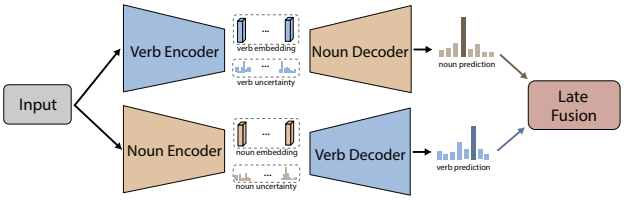
[*] Work done during internship at Honda Research Institute, USA.

ferent outputs from the same input. The outputs are used to quantify the predictive uncertainty. In general, we stack two encoder layers to form both the verb encoder and noun encoder. To improve efficiency, only the last layer needs sampling. The output embedding of the last layer are used to compute $\mathcal{L}_{feat}$. Meanwhile, each output embedding goes through a fully connected network to make action anticipation at each time step.

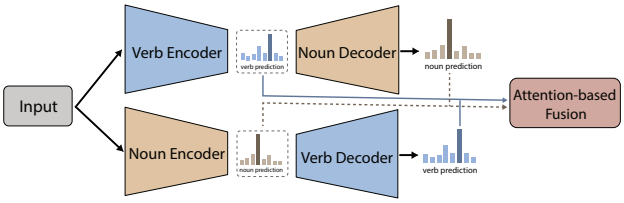
**Decoders.** The verb/noun decoder contains a cross-attention layer and self-attention layers. The cross-attention layer aims to leverage the embeddings generated by the encoders to help the anticipation. The cross-attention layer takes both the initial features $\boldsymbol{F}_t$ extracted by the backbone and the embeddings $\boldsymbol{Z}_{nt}/\boldsymbol{Z}_{vt}$ from the encoder and exchange the information with each other. Uncertainty masks are multiplied to $\hat{\boldsymbol{F}}_t$ to filter out redundant and irrelevant information. In this work, we only use one cross-attention layer to incorporate information from the encoders. Embeddings from the future at each time step are masked to ensure the anticipation property. After the cross-attention layer, we stack two self-attention layers that are identical to the encoder. The output embeddings from the final self-attention layer are used to compute $\mathcal{L}_{feat}$ of the decoder and action anticipation at each time step.




(a) Early Fusion




(b) Late fusion




(c) Attention-based fusion

Figure A1. **Illustration of fusion strategies.**

### 1.3. Fusion strategies

In the ablation study part of the main paper, we implemented a few fusion strategies for comparison including early fusion, late fusion, and attention-based fusion. The overall frameworks of them are shown in Figure A1. Here we include the implementations details of each strategy.

**Early fusion.** The early fusion simply combines the predictions from the verb encoder and noun encoder. Since the encoders also make verb/noun anticipation at each time step, the prediction of next $(verb, noun)$ pair is available before finishing the forward process through decoders. Specifically, the verb prediction from the verb-to-noun encoder and noun prediction from the noun-to-verb encoder are combined as the action anticipation. We also perform the same post-processing procedures as UADT to correct implausible $(verb, noun)$ pairs.

**Late fusion.** The late fusion combines the initial predictions from the verb decoder and noun decoder. Specifically, the noun prediction of the verb-to-noun model and verb prediction of the noun-to-verb prediction are combined as the action prediction. The post-processing process is also applied to late fusion.

**Attention-based fusion** [6]. The early fusion and late fusion are relatively rigid. We followed the design as the modality attention in [6]. A fully-connected network is jointly trained with the model to select the predictions from both verb-to-noun and noun-to-verb model. Specifically, learnable attention masks are applied on element-wise product. The noun predictions of the verb-to-noun decoder and noun-to-verb decoder are combined. The verb predictions of the noun-to-verb decoder and verb-to-noun encoder are combined. By applying the attention, the model automatically identifies reliable predictions from both models. We also perform the post-processing since the attention-based fusion may also output implausible $(verb, noun)$ pairs.

## 2. Computational Efficiency

Computational efficiency is an important factor of deploying action anticipation algorithms to real-world applications. In this section, we discuss the computational efficiency of UADT (§ 2.1), the training cost (§ 2.2), and uncertainty sampling to latency (§ 2.3).

### 2.1. Efficiency

An efficiency comparison is shown in Table A1. From the comparison, UADT has less model parameters and inference latency than most methods. The inference latency includes both the feature extraction time of backbone and inference time of model.

| Method | Init | Training | # Parameters ($\times 10^6$) | Inference Latency (ms) | Top-5 Recall | | |
|---|---|---|---|---|---|---|---|
| | | | | | Verb | Noun | Action |
| AVT [8] | IN21k | - | 378 | 420 | 30.2 | 31.7 | 14.9 |
| MeMViT [15] | K400 | - | 59 | 160 | 32.8 | 33.2 | 15.1 |
| RAFTformer [7] | K400+IN1k | - | 26 | 40 | 33.3 | 35.5 | 17.6 |
| UADT (ours) | K400 | Two-stage | 47 | 105 | 35.2 | 38.5 | 18.8 |
| UADT (ours) | K400 | E2E-one-stage | 47 | 105 | 37.3 | 40.1 | 19.3 |
| UADT (ours) | K400 | E2E-two-stage | 47 | 105 | **37.4** | **40.4** | **19.5** |
| MeMViT [15] | K700 | - | 212 | 350 | 32.2 | 37.0 | 17.7 |
| RAFTformer [7] | K700 | - | 26 | 110 | 33.7 | 37.1 | 18.0 |
| RAFTformer-2B [7] | K700+IN1k | - | 52 | 160 | 33.8 | 37.9 | 19.1 |
| UADT (ours) | K700 | Two-stage | 47 | 127 | 38.2 | 41.4 | 20.3 |
| UADT (ours) | K700 | E2E-one-stage | 47 | 127 | 41.2 | 42.8 | 21.1 |
| UADT (ours) | K700 | E2E-two-stage | 47 | 127 | **41.5** | **43.0** | **21.2** |

Table A1. **Efficiency comparison on EK100.** UADT maintains relatively small number of model parameters compared to most methods. Inference latency is measured on a Nvidia Tesla V100 GPU.

| Method | Init | Training | GPU Hours |
|---|---|---|---|
| UADT | K400 | Two-stage | 31 |
| UADT | K400 | E2E-one-stage | 52 |
| UADT | K400 | E2E-two-stage | 46 |
| UADT | K700 | Two-stage | 35 |
| UADT | K700 | E2E-one-stage | 53 |
| UADT | K700 | E2E-two-stage | 48 |

Table A2. **Training cost comparison on EK100.** The end-to-end (E2E) training results in better performance but requires more training time.

## 2.2. Training cost

In the main paper, we show that UADT can be trained by a two-stage strategy of end-to-end training. We make a training cost comparison in Table A2. The GPU hours are reported on a Nvidia RTX 3090 Ti GPU. Compared to most other methods, UADT has a relatively small number of model parameters and training cost. Although UADT and RAFTformer [7] have similar transformer-based architectures, UADT costs more training times due to the uncertainty sampling. In addition, end-to-end (E2E) training has higher training cost and better performance. The reason of longer training time is that E2E training also trains the encoders and takes more time to converge.

## 2.3. Uncertainty sampling to latency

UADT requires sampling to quantify the uncertainty, insufficient sampling may affect the accuracy of uncertainty estimation and further degrade the anticipation performance. On the other hand, increasing the number of samples causes more computation cost. To find the optimal trade-off, we study the relationship among performance, samples times, and inference latency. The comparison is plotted in Figure A2. From the plot, the latency increases as the number of samples increases since the forward process after the sampling are repeated for more times.

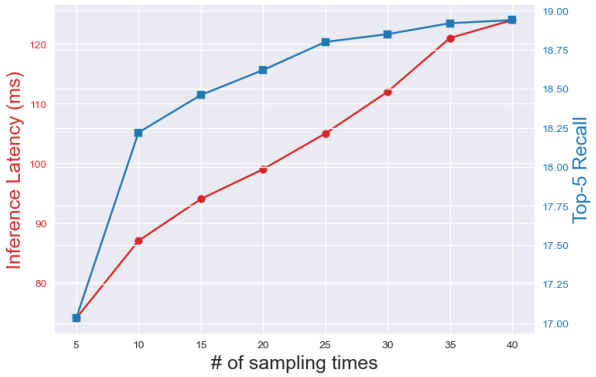



Figure A2. **Uncertainty sampling and latency**. The latency increases as more sampling are performed. Latency is measured on a 16G Nvidia Tesla V100 GPU with K400 features.

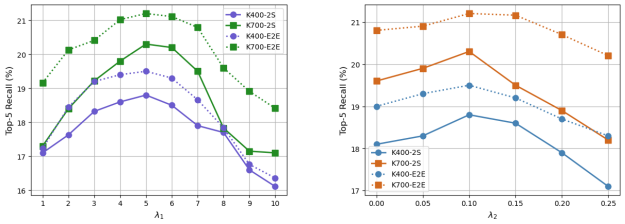
# 3. Ablation Study of Decoder Loss

The total decoder loss function (Eq. 1) contains three terms: $\mathcal{L}_{next}$ for next verb/noun anticipation, $\mathcal{L}_{feat}$ for feature anticipation, and $\mathcal{L}_{verb/noun}$ for past verb/noun anticipation. To obtain the optimal hyper-parameters $\lambda_1$ and $\lambda_2$, we did a ablation study and results are shown in Figure A3. Based on the experiment results, we empirically choose $\lambda_1 = 5$ and $\lambda_2 = 0.1$ since this setting generates the best performance. The abation also demonstrates the necessity of $\mathcal{L}_{verb/noun}$ since the performance drops if we set $\lambda_2 = 0$.

$$\mathcal{L}_{de} = \mathcal{L}_{next} + \lambda_1 \mathcal{L}_{feat} + \lambda_2 \mathcal{L}_{verb/noun} \quad (1)$$

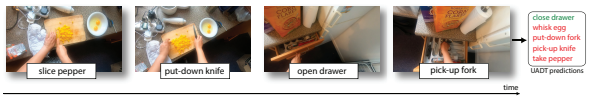
# 4. Qualitative Results

Here, we provide a few qualitative examples of UADT action anticipation on different datasets in Figure A4.

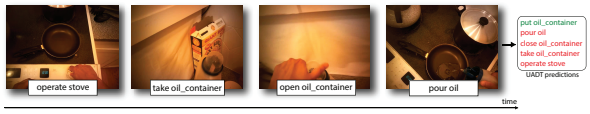



(a) $\lambda_1$ (b) $\lambda_2$

Figure A3. **Ablation study of decoder loss function on EK100 val.** We set $\lambda_1 = 5$ and $\lambda_2 = 0.1$ since they give the best performance.




(a) EK100




(b) EGETA Gaze+


(c) 50-Salads

Figure A4. **Qualitative examples of UADT prediction.** The green $(verb, noun)$ pair indicates the prediction and red ones are incorrect predictions.

**Algorithm A1** Inference workflow

**Input:** $\{\boldsymbol{X}_t \in \mathbb{R}^{t\times H\times W\times C}\}$: test input
**Output:** $\{\hat{y}_{t+t_f}\}$: predicted action label

Extract $\boldsymbol{F}_t$ from $\boldsymbol{X}_t$ by a backbone
Make predictions $\hat{\boldsymbol{V}}_t$ and $\hat{\boldsymbol{N}}_t$
Compute anticipated features $\hat{\boldsymbol{F}}_t$
Compute encoder uncertainty $\mathcal{U}_e$ by Eq. 5
Compute uncertainty-masked features $\hat{\boldsymbol{F}'}_t$
Generate intermediate embeddings $\boldsymbol{Z}_{nt}$ and $\boldsymbol{Z}_{vt}$
Cross-attention between $\boldsymbol{Z}_{nt}/\boldsymbol{Z}_{vt}$ and $\hat{\boldsymbol{F}'}_t$
Predict $\boldsymbol{V}_t$ and $\boldsymbol{N}_t$
Post-processing by Eq. 13
**return** $\{\hat{y}_{t+t_f}\}$

## 5. Inference Workflow

During the inference, parameters besides the probabilistic feed forward networks are fixed and the input go through the encoders and decoders to generate the output. The inference workflow is summarized as Algorithm A1.

| Architecture | Top-5 Recall | | |
|---|---|---|---|
| | Verb | Noun | Action |
| Decoder [53] | 33.2 | 34.9 | 17.1 |
| cross-self | 33.1 | 34.4 | 16.9 |
| self-cross | 34.1 | 36.9 | 18.0 |
| self-cross-self | **35.2** | **38.5** | **18.8** |

Table A3. **Ablation study of decoder architecture.** The results are obtained with K400 features.

| Method | Overall | | | Unseen Kitchen | | | Tail Classes | | |
|---|---|---|---|---|---|---|---|---|---|
| | Verb | Noun | Act | Verb | Noun | Act | Verb | Noun | Act |
| AVT++ | 25.6 | 28.8 | 12.6 | 20.9 | 22.3 | 8.8 | 19.0 | 22.0 | 10.1 |
| [67] | 20.7 | 31.8 | 14.9 | 16.2 | 27.7 | 12.1 | 13.4 | 23.8 | 11.8 |
| [25] | 30.1 | 34.1 | 15.4 | - | - | - | - | - | - |
| UADT-b | 35.8 | 37.6 | 17.0 | 33.8 | 36.9 | 16.6 | 31.6 | 32.7 | 12.9 |
| UADT | **40.3** | **43.1** | **18.2** | **39.9** (+6.1) | **42.4** (+5.5) | **17.3** | **36.8** (+5.2) | **35.0** (+2.3) | **14.3** |

Table A4. **Experiment results on EK100 test set with K700 RGB features.** "b" denotes base version without uncertainty modeling.

## 6. Ablation Study of Decoder Architecture Design

For the self-cross-self design in the decoder, we replaced it with self-cross, cross-self, and original transformer decoder. A comparison on EK100 is shown in Table A3. The original transformer decoder makes the prediction in an autoregressive way, which may not be ideal for incorporating encoder information. Also, the self attention is necessary before the cross attention as the input features need to be mapped to a subspace that is close to the encoder output.

## 7. Experiment Results on EK100 Test set

We include the results on EK100 test set with K700 RGB features in Table A4. Our UADT outperforms both its base version without uncertainty modeling and prior works, hence demonstrating the effectiveness of our model especially for improving verbs and nouns in unseen and low-data regimes, aligning with our uncertainty modeling objectives.

## 8. Effectiveness of Cross-Attention

We trained the model solely for verb/noun prediction without cross attention. The results on EK100 are shown in Table A5. We remove the uncertainty modeling in NtV and VtN for fair comparisons. The NtV/VtN outperforms the Verb/Noun model, which indicates the cross-information helps the task.

## 9. Uncertainty Modeling

In general, our novelty lies in how to adapt uncertainty for anticipation task. First, [10] models the distributions of attention scores of transformer because it aims at capturing the dependencies among sub-actions of a complex action.

| Method | Top-5 Recall | | |
|---|---|---|---|
| | Verb | Noun | Action |
| verb-only | 30.1 | - | - |
| noun-to-verb | **31.7** | - | - |
| noun-only | - | 31.3 | - |
| verb-to-noun | - | **35.5** | - |
| verb+noun | - | - | 14.9 |
| NtV+VtN | 33.2 | 36.5 | **18.0** |

Table A5. **Comparison with no cross-attention models.**

| Method | Init | Top-5 Recall | | |
|---|---|---|---|---|
| | | Verb | Noun | Action |
| UADT+[10] | K400 | 34.5 | 37.3 | 18.1 |
| UADT | | **35.2** | **38.5** | **18.8** |
| UADT+[10] | K700 | 36.3 | 40.0 | 19.4 |
| UADT | | **38.2** | **41.4** | **20.3** |

Table A6. **Comparison with uncertainty modeling in [29].**

Differently, we model the distributions of parameters of the feed-forward layers to model all past observation for action anticipation. Second, the uncertainty in [10] is used to weigh training data based on their frequency to balance the training. In our work, the uncertainty is mainly used for selecting reliable features from the encoder to better serve the decoder. We replaced the attention layers of UADT with score-based layers in [10] and compare with the original one with RGB features on EK100-val. The results in Table A6 demonstrate our uncertainty modeling works better for anticipation task.

## References


[1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 1

[2] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019. 1

[3] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European conference on computer vision (ECCV)*, pages 720–736, 2018. 1

[4] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6824–6835, 2021. 1

[5] Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3575–3584, 2019. 1

[6] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6252–6261, 2019. 1, 2

[7] Harshayu Girase, Nakul Agarwal, Chiho Choi, and Karttikeya Mangalam. Latency matters: Real-time action forecasting transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18759–18769, 2023. 3

[8] Rohit Girdhar and Kristen Grauman. Anticipative video transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13505–13515, 2021. 3

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1

[10] Hongji Guo, Hanjing Wang, and Qiang Ji. Uncertainty-guided probabilistic transformer for complex action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20052–20061, 2022. 4, 5

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[12] Debaditya Roy and Basura Fernando. Action anticipation using latent goal learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2745–2753, 2022. 1

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

[14] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016. 1

[15] Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13587–13597, 2022. 3

[16] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition: 29th DAGM Symposium, Heidelberg, Germany, September 12-14, 2007. Proceedings 29*, pages 214–223. Springer, 2007. 1