⋆ Supplementary Material ⋆ *Ada*MAE: Adaptive Masking for Efficient Spatiotemporal Learning with Masked Autoencoders

A. Pre-training on smaller datasets

Table 1 compares the performance of *Ada*MAE on smaller datasets such as UCF101 and HMDB51 for the default configuration in Table 1. We can see that the proposed *Ada*MAE performs well on smaller datasets as well.

Method	UCF	HMDB
VideoMAE	91.3	62.6
*Ada*MAE	**92.8**	**64.0**

Table 1. Results on smaller datasets.

B. Feature transferability

we present fine-tuning results on smaller datasets, UCF and HMDB for the model pre-trained on K400 in Tab. 2, in addition to the already discussed transfer learning results on SSv2 and K400 datasets in the paper. These results empirically demonstrate that representations learned by *Ada*MAE are more transferable than VideoMAE and STMAE.

Method	SSv2	UCF	HMDB
VideoMAE	68.5	96.1	73.3
STMAE	69.1	NA	NA
*Ada*MAE	**69.9**	**97.0**	**74.1**

Table 2. Transfer learning performance (top-1 %) of pre-trained model on K400.

C. Linear probing results

We compare linear probing results of *Ada*MAE with VideoMAE and STMAE in Tab. 3 for the exact setting used in VideoMAE. As shown in Tab. 3, *Ada*MAE achieves the best top-1 accuracy with lower budget supervision which empirically shows that representations learned by *Ada*MAE are more favorable towards action classification than VideoMAE.

Method	SSv2
VideoMAE	38.9
STMAE	NA
*Ada*MAE	**40.1**

Table 3. Linear probing results (top-1 %).

D. Computational efficiency:

We benchmark average time per epoch of *Ada*MAE and VideoMAE on the SSv2 dataset for the ViT-B backbone with batch size of (16 /gpu) $\times$ 8 gpus. We can see that the proposed *Ada*MAE completes one epoch 38 s faster than VideoMAE / STMAE. If we pre-train for 800 epochs, then this results in saving 38 s $\times 800 \approx 506$ min ≈ 8 hrs!, while utilizing lower memory (see ablation study) and achieving better performance on the downstream tasks.

Method	Avg. Time / Epoch
VideoMAE	9 min 49 s
*Ada*MAE	**9** min **11** s

Table 4. Wall clock time.

Memory Efficiency: The slight memory increase (+0.7 GB) at the 90% masking ratio is due to the sampling network in our architecture. However, this sampling network enables us to achieve better performance with a higher masking ratio (95%) which ultimately reduces the memory requirement compared to VideoMAE / STMAE.

E. Pseudocode for *Ada*MAE

The pseudocode for our `AdaptiveTokenSampler` network is given in Algorithm 1 and *Ada*MAE for 2.

Algorithm 1 Pseudo-code for AdaptiveTokenSampler.

```
Inputs: Tokenized video: $\boldsymbol{X} \in \mathbb{R}^{N\times d}$, Masking ratio: $\rho \in (0,1)$
Outputs: Categorical distribution: $\boldsymbol{p}$, Sampled mask indices: $\boldsymbol{I}_m$, Sampled mask: $\boldsymbol{M}$
$P$ = TokenProbs($\boldsymbol{X}$)                    ▷ token probabilities
$\boldsymbol{p} \sim$ CAT($\boldsymbol{P}$)                      ▷ categorical distribution
$N_v$ = int($N \times (1-\rho)$)              ▷ # of visible tokens
$\boldsymbol{I}_v$ = $\boldsymbol{p}$.sample($N_{\text{vis}}$)               ▷ visible token indices
$\boldsymbol{I}_m = \boldsymbol{U} - \boldsymbol{I}_v$                        ▷ mask token indices
$\boldsymbol{M}$ = GetBooleanMask($\boldsymbol{I}_v, \boldsymbol{I}_m$)          ▷ binary mask
```

Algorithm 2 Pseudo-code for our *Ada*MAE.

```
Input: The video dataset $\mathcal{D} = \{\boldsymbol{V}_i$ : for $i = 1,2,3,\cdots,|\mathcal{D}|\}$, Masking ratio: $\rho \in (0,1)$
for $\boldsymbol{V}_i \in \mathcal{D}$ do
    $\boldsymbol{X}_i$ = Tokenizer($\boldsymbol{V}_i$)                      ▷Tokens
    $\boldsymbol{X}_i = \boldsymbol{X}_i$ + PosEmbd             ▷Positional embedding
    $\boldsymbol{p}, \boldsymbol{I}_v, \boldsymbol{M}$ = AdaptiveTokenSampler($\boldsymbol{X}_i, \rho$)
    $\boldsymbol{X}_v = \boldsymbol{X}_i[\sim \boldsymbol{M}]$                        ▷ visible tokens
    $\boldsymbol{F}_v$ = ViTEncoder($X_v$)                   ▷ visible feats
    $\boldsymbol{F}_v = \boldsymbol{F}_v$ + PosEmbd[$\sim \boldsymbol{M}$]
    $\boldsymbol{F}_m = f_m$ + PosEmbd[$\boldsymbol{M}$]                    ▷ mask feats
    $\boldsymbol{F} = \boldsymbol{F}_v \oplus \boldsymbol{F}_m$                     ▷ visible + mask feats
    $\widehat{\boldsymbol{X}}$ = ViTDecoder($\boldsymbol{F}$)                            ▷Decoder
    $\widehat{\boldsymbol{X}}_m = \widehat{\boldsymbol{X}}[\boldsymbol{M}]$                        ▷ masked prediction
    $\boldsymbol{X}_m = \boldsymbol{X}[\boldsymbol{M}]$                               ▷ masked GT
    $\mathcal{L}_R = \|\widehat{\boldsymbol{X}}_m - \boldsymbol{X}_m\|_2$                ▷ reconstruction loss
    $\mathcal{L}_S = -(\boldsymbol{p}.\text{LogProb}(\boldsymbol{I}_m)) \cdot \mathcal{L}_R.detach()$   ▷ sampling loss
    $\mathcal{L} = \mathcal{L}_R + 1e-4 \cdot \mathcal{L}_S$                        ▷ final loss
    $\mathcal{L}.backward()$                          ▷ back-propagation
end for
```

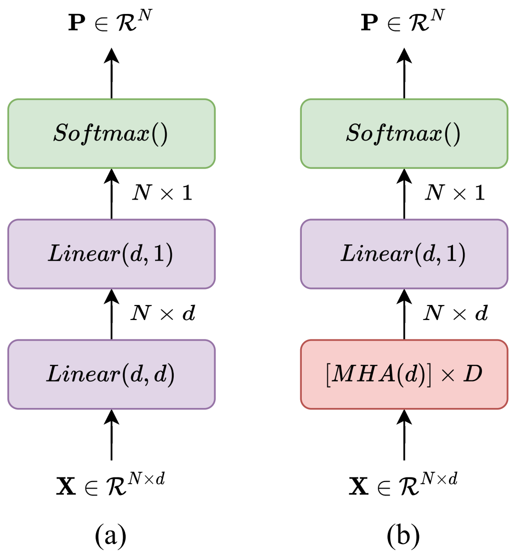
F. Network architectures for Adaptive Token Sampler

We consider two network architectures for our adaptive sampling network.

1. **MLP:** Keeping the importance of a computationally light-weight design for an adaptive sampling network, we first experimented with a simple MLP network for the sampling network as shown in Fig. 1 (a). In this architecture, we process all the tokens ($\mathbf{X} \in \mathbb{R}^{N\times d}$) through a Linear layer with in_features = out_features = d (denoted by Linear(d, d)) followed by a Linear(d, 1) to bring down the embedding dimension from d to 1. We then apply a Softmax() layer to obtain the probability values $\mathbf{P} \in \mathbb{R}^N$.

   Although this network is computationally very light-weight, the experimental results were not encouraging. Specially, it was not able to capture the spatiotemporal information of input tokens; resulted in predicted probability density map to have no/very-less relationship with the high/low activity regions. This resulted in poor performance as demonstrated in ablation study. This is understandable because it operates only on the embedding dimension and not be able to model the interaction between the patches that is necessary for predicting proper probability map.

2. **MHA:** Motivated from the drawback that we observed with MLP-based sampling network, we decided to utilize multi-head self-attention network for this purpose, because it can properly model the interconnectivty between all the patches through the attention mechanism. Since we want to keep it computationally light-weight as much as possible, we experimented with the effect of different number of blocks and reduced embedding dimension. By utilizing MHA-based sampling network, we were able to generate visually appealing probability density maps (as shown in Figure 2-6) that are in line with our intuition for adaptive sampling.



$\mathbf{P} \in \mathcal{R}^N$
$Softmax()$
$N \times 1$
$Linear(d,1)$
$N \times d$
$Linear(d,d)$
$\mathbf{X} \in \mathcal{R}^{N\times d}$
(a)
$\mathbf{P} \in \mathcal{R}^N$
$Softmax()$
$N \times 1$
$Linear(d,1)$
$N \times d$
$[MHA(d)] \times D$
$\mathbf{X} \in \mathcal{R}^{N\times d}$
(b)

Figure 1. **The network architectures considered for adaptive token sampling network: (a)** MLP-based sampling network **(b)** MHA-based sampling network.

F.1. Understanding the formulation of sampling loss L_S

Here we provide some additional visualizations to further understand our *Ada*MAE for selected videos from SSv2 (Figure 2, 3, 4, 5, and 6) and K400 (Figure 7, 8, 9, and 10.).

For example, let us consider Figure 2. We see that most of the spatiotemporal information is concentrated in the upper part of each frame (the region containing the bottle and the part of a hand), making those patches difficult to reconstruct accurately (see second and third rows). Since the proposed `AdaptiveSampling` network is optimized by maximizing $\mathbb{E}[L_R]$, it predicts higher probabilities for the patches from the high activity region. Hence, when sampling the visible tokens, MAE gets relatively more tokens from the high activity region than the low activity region as shown in the last row.

We can make the similar observations for the other examples as well.

Note that these results are for the default setting except here we visualize at 100-th epoch of pre-training.

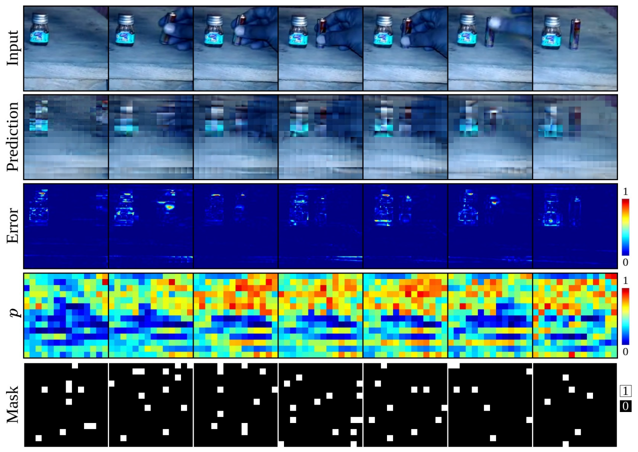


Figure 2. An example visualization of our adaptive sampling on SSv2 dataset.

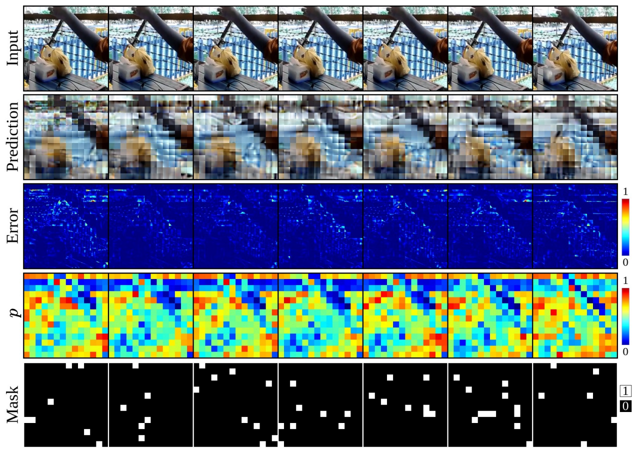


Figure 3. An example visualization of our adaptive sampling on SSv2 dataset.

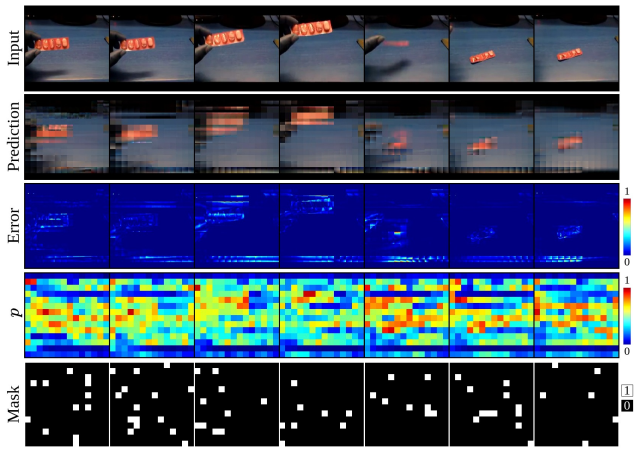


Figure 4. An example visualization of our adaptive sampling on SSv2 dataset.

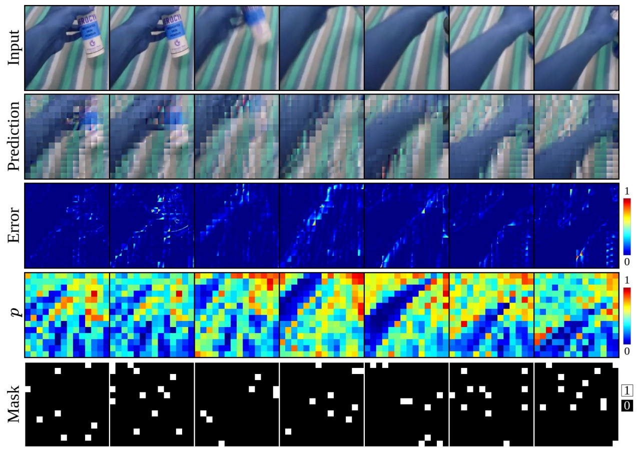


Figure 5. An example visualization of our adaptive sampling on SSv2 dataset.

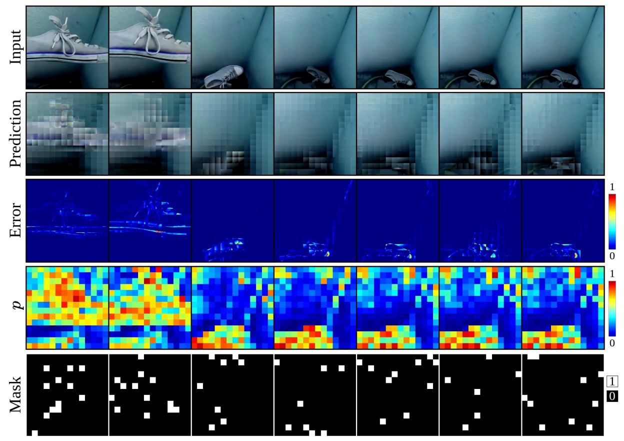


Figure 6. An example visualization of our adaptive sampling on SSv2 dataset.

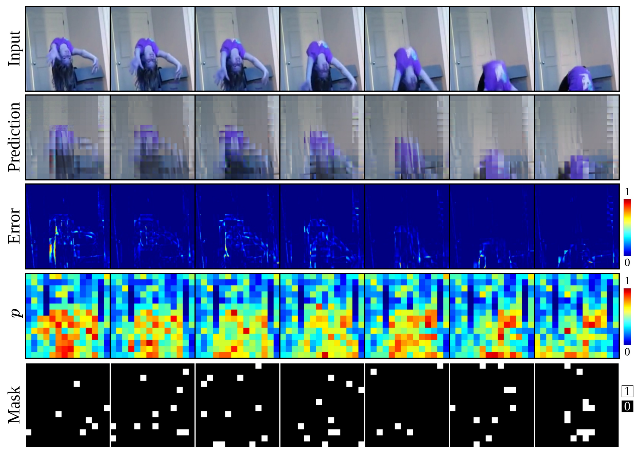


Figure 7. An example visualization of our adaptive sampling on K400 dataset.

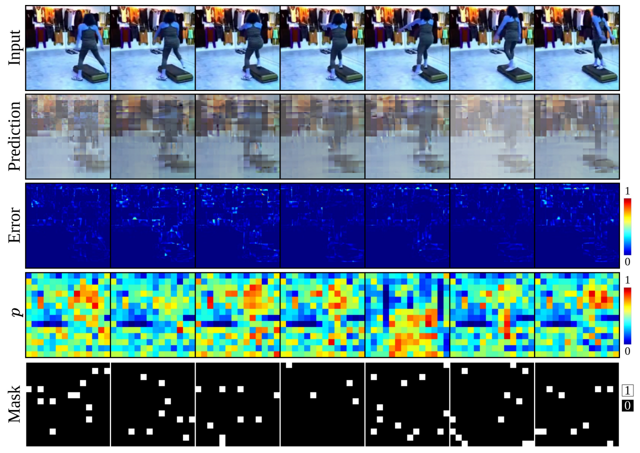


Figure 8. An example visualization of our adaptive sampling on K400 dataset.

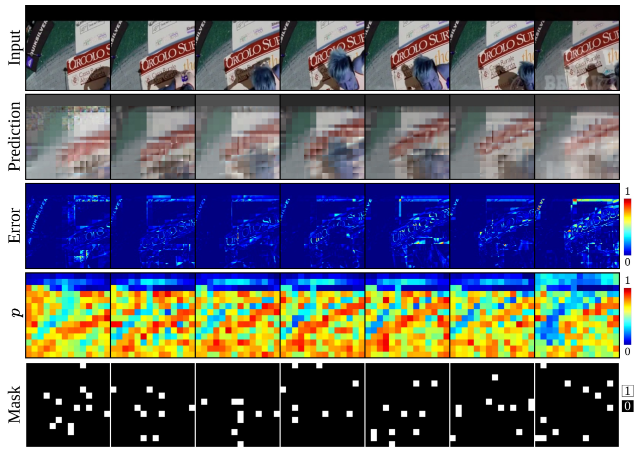


Figure 9. An example visualization of our adaptive sampling on K400 dataset.

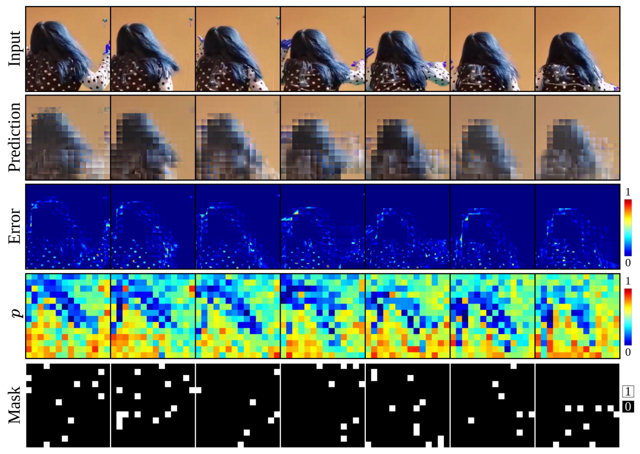


Figure 10. An example visualization of our adaptive sampling on K400 dataset.

G. Encoder-decoder architecture

Table 5 summarizes the encoder-decoder architecture of our *Ada*MAE. Specifically, we consider 16-frame vanilla ViT-Base architecture for all experiments. We use asymmetric encoder-decoder architecture for self-supervised pre-training and discard the decoder during the fine-tuning.

Given a video we first extract 16 frames ($3 \times 16 \times 224 \times 224$). We use temporal stride of 4 and 2 for K400 and SSv2 datasets, respectively. Next we process these 16 frames through a Tokenizer which is essentially a 3D convolution layer with kernel size of $2 \times 16 \times 16$, stride of $2 \times 16 \times 16$ and output embedding dimension of 768. This process results in a total of 1568 tokens and each token is represented by a 768 dimension vector. Next, we sample $(1 - \rho) \times 1568$ number of tokens as the visible tokens from our adaptive token sampling network. These visible tokens are then processed through the ViT encoder that comprises of 12 cascaded multi-head self-attention blocks (MHA blocks). The outputs from the ViT encoder is then combined with a fixed learnable representation for masked tokens which results in the 1568 token representations. These 1568 representations are then process through a projector which brings down their embedding dimension from 768 embedding dimension to 384 by an MLP layer. These projected representations are then process through the ViT-decoder which consists of 4 MHA blocks followed by an MLP layer to bring the embedding dimension from 384 to the total number of pixels in a cube which given by $2 \times 3 \times 16 \times 16 = 1536$. This is finally reshaped back back to the original space and used to compute the reconstruction loss.

Table 5. Encoder-Decoder architecture of our *Ada*MAE. MHA denotes joint space-time multi-head self-attention.

Stage	ViT-Base	Output shape
Input Video	stride $4 \times 1 \times 1$ for K400 stride $2 \times 1 \times 1$ for SSv2	$3 \times 16 \times 224 \times 224$
Tokenization	stride $2 \times 16 \times 16$ emb. dim 768	1568×768
Masking	Adaptive Masking masking ratio ρ	$[(1-\rho) \times 1568] \times 768$
Encoder	$[MHA(768)] \times 12$	$[(1-\rho) \times 1568] \times 768$
Projection	$MHA(384)$ concat masked tokens	1568×384
Decoder	$[MHA(384)] \times 4$	$[(1-\rho) \times 1568] \times 384$
Projector	$MLP(1536)$	1568×1536
Reshaping	from 1536 to $3 \times 2 \times 16 \times 16$	$3 \times 16 \times 224 \times 224$

H. Pre-training setting

Table 6 summarizes the pre-training setting on SSv2 and K400 dataset. In addition, we linearly scale the base learning rate with respect to the overall batch size, lr = base learning rate× batch size 256. We adopt the PyTorch and DeepSpeed frameworks for faster training.

Table 6. Pre-training setting on SSv2 and K400 datasets.

Configuration	SSv2	K400
Optimizer	Adamw	
Optimizer betas	{0.9, 0.95}	
Base learning rate	1.5e-4	
Weight decay	5e-2	
Learning rate shedule	cosine decay	
Warmup epochs	40	
Flip augmentation	False	True
Augmentation	MultiScaleCrop	

I. End-to-end fine tuning setting

Table 7 summarizes the end-to-end fine-tunining setting on SSv2 and K400 datasets.

J. Mask visualizations

Figure 11, 12, 13, and 14 show the adaptive mask visualizations for masking ratio of 80%, 85%, 90%, and 95% for selected videos from SSv2 dataset.

K. Animated mask visualizations.

The animated mask visualizations of our adaptive masking technique can be found through these links:

1. *For SSv2 dataset:* `https://www.dropbox.com/s/z0ijog5vt5sa546/adamae_vis_ssv2.mp4`

Table 7. Pre-training setting on SSv2 and K400 datasets.

Configuration	SSv2	K400
Optimizer	Adamw	
Optimizer betas	{0.9, 0.999}	
Base learning rate	1.5e-4	1e-3
Layer-wse lr decay	0.75	
Weight decay	5e-2	
Learning rate shedule	cosine decay	
Warmup epochs	40	
Flip augmentation	False	True
RandAug	(0.9, 0.05)	
Label smoothing	0.1	
Mixup	0.8	
Cutmix	1.0	
drop path	0.1	

2. *For Kinetics-400:* `https://www.dropbox.com/s/eki7q829x4xo70c/adamae_vis_k400.mp4`

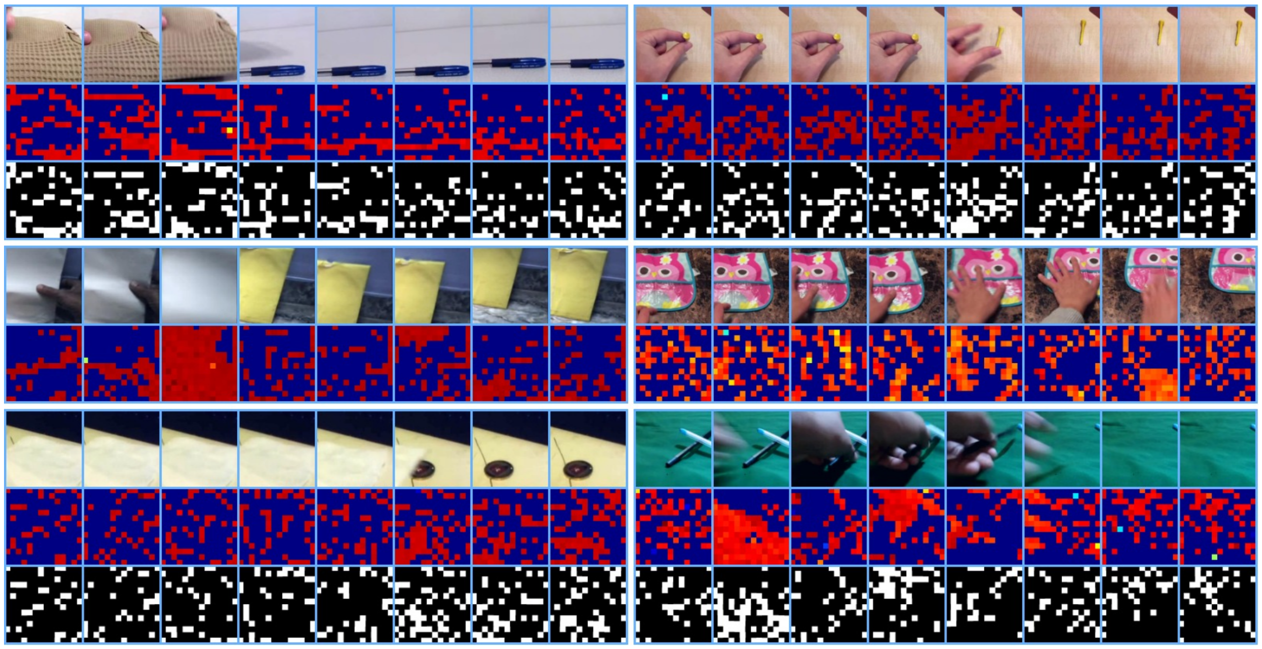


Figure 11. Mask visualizations of our *Ada*MAE for **80% masking ratio** on SSV2 dataset.Given a video (*first row*), our *Ada*MAE first predicts the categorical distribution (*second row*), and then sample the mask (*third row*) from that distribution. Colors closer **red** and **blue** denotes the patches with *high* and *low* probability, respectively. In mask visualizations, **black** and **white** corresponds to the masked and visible path locations, respectively.

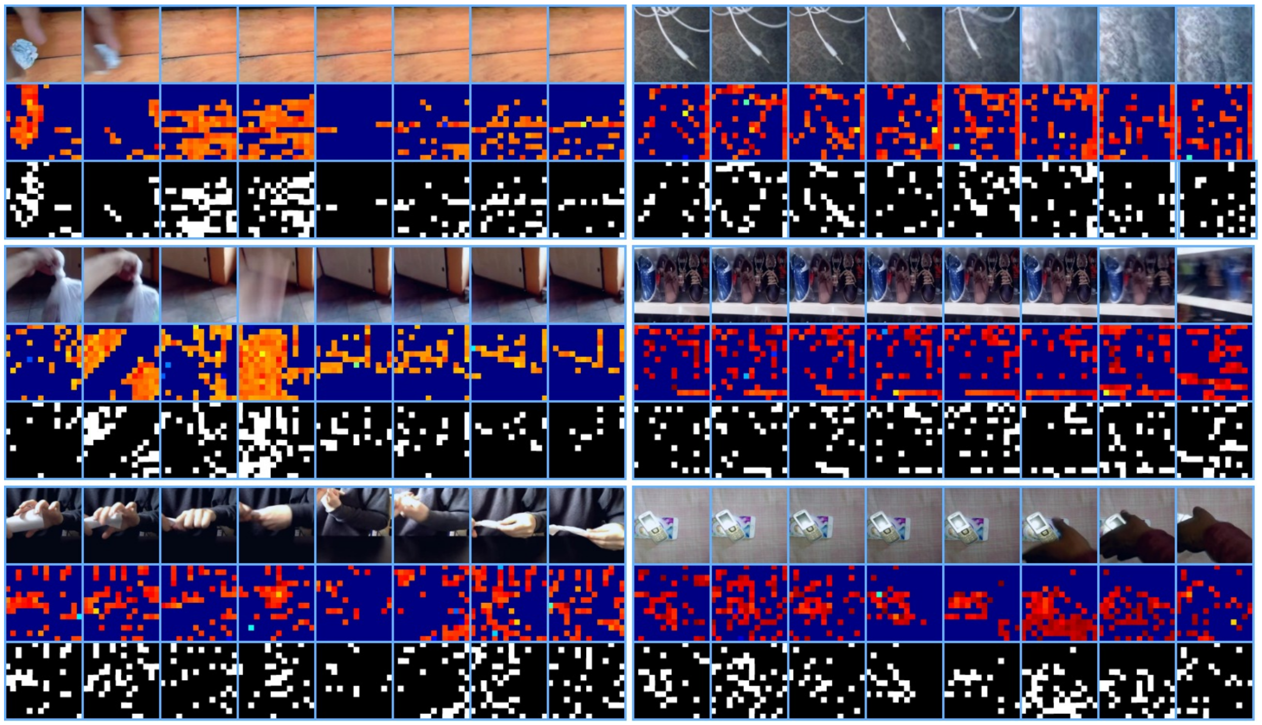


Figure 12. Mask visualizations of our *Ada*MAE for **85% masking ratio** on SSV2 dataset.Given a video (*first row*), our *Ada*MAE first predicts the categorical distribution (*second row*), and then sample the mask (*third row*) from that distribution. Colors closer **red** and **blue** denotes the patches with *high* and *low* probability, respectively. In mask visualizations, **black** and **white** corresponds to the masked and visible path locations, respectively.

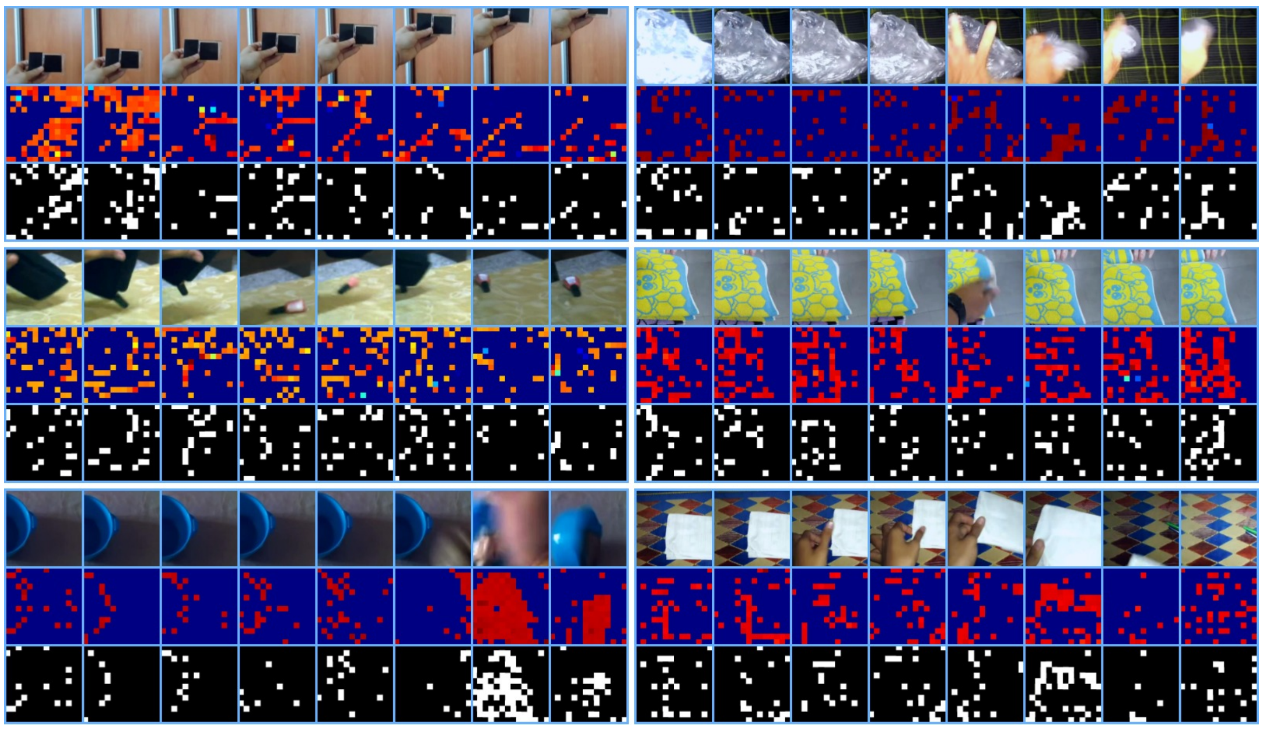


Figure 13. Mask visualizations of our *Ada*MAE for **90% masking ratio** on SSV2 dataset.Given a video (*first row*), our *Ada*MAE first predicts the categorical distribution (*second row*), and then sample the mask (*third row*) from that distribution. Colors closer **red** and **blue** denotes the patches with *high* and *low* probability, respectively. In mask visualizations, **black** and **white** corresponds to the masked and visible path locations, respectively.

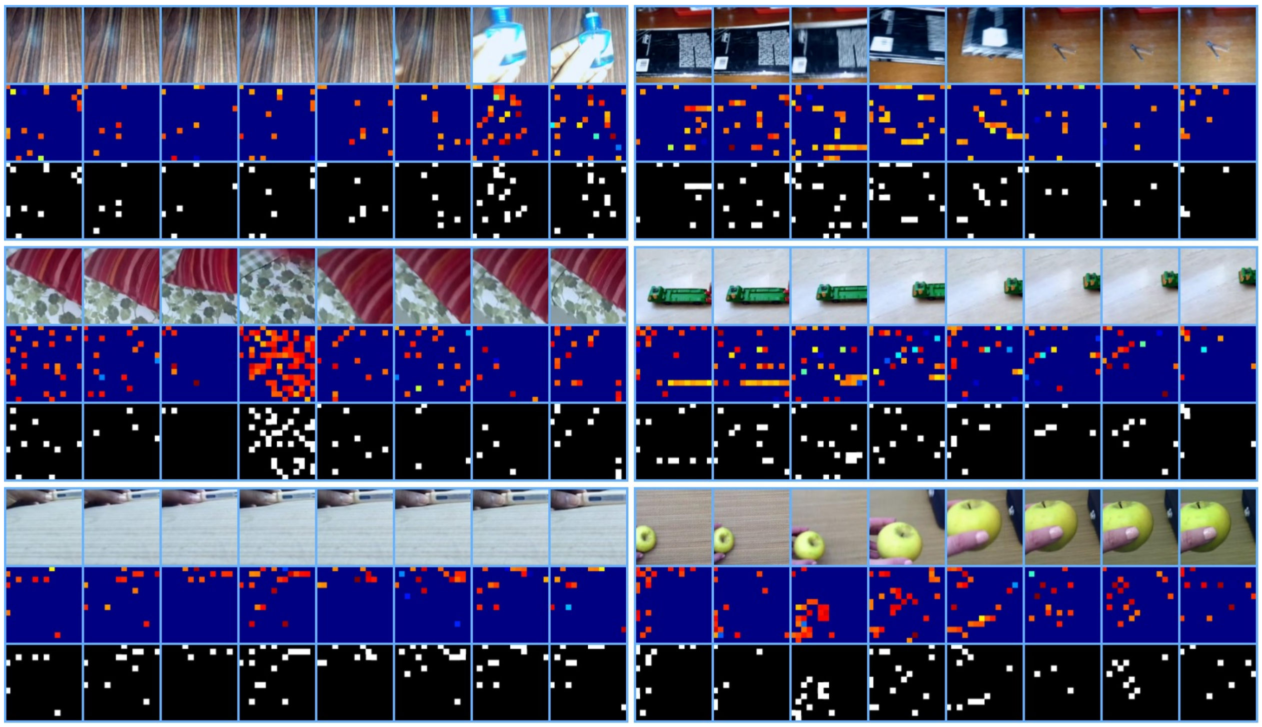


Figure 14. Mask visualizations of our *Ada*MAE for **95% masking ratio** on SSV2 dataset.Given a video (*first row*), our *Ada*MAE first predicts the categorical distribution (*second row*), and then sample the mask (*third row*) from that distribution. Colors closer **red** and **blue** denotes the patches with *high* and *low* probability, respectively. In mask visualizations, **black** and **white** corresponds to the masked and visible path locations, respectively.