BEVT: BERT Pretraining of Video Transformers

Rui Wang\textsuperscript{1,2}, Dongdong Chen\textsuperscript{3}, Zuxuan Wu\textsuperscript{1,2,*}, Yinpeng Chen\textsuperscript{3}, Xiyang Dai\textsuperscript{3}, Mengchen Liu\textsuperscript{3}, Yu-Gang Jiang\textsuperscript{1,2,*}, Luowei Zhou\textsuperscript{3}, Lu Yuan\textsuperscript{3}

\textsuperscript{1}Shanghai Key Lab of Intelligent Info. Processing, School of Computer Science, Fudan University
\textsuperscript{2}Shanghai Collaborative Innovation Center on Intelligent Visual Computing
\textsuperscript{3}Microsoft Cloud + AI

Abstract

This paper studies the BERT pretraining of video transformers. It is a straightforward but worth-studying extension given the recent success from BERT pretraining of image transformers. We introduce BEVT which decouples video representation learning into spatial representation learning and temporal dynamics learning. In particular, BEVT first performs masked image modeling on image data, and then conducts masked image modeling jointly with masked video modeling on video data. This design is motivated by two observations: 1) transformers learned on image datasets provide decent spatial priors that can ease the learning of video transformers, which are often times computationally-intensive if trained from scratch; 2) discriminative clues, i.e., spatial and temporal information, needed to make correct predictions vary among different videos due to large intra-class and inter-class variations. We conduct extensive experiments on three challenging video benchmarks where BEVT achieves very promising results. On Kinetics 400, for which recognition mostly relies on discriminative spatial representations, BEVT achieves comparable results to strong supervised baselines. On Something-Something-V2 and Diving 48, which contain videos relying on temporal dynamics, BEVT outperforms by clear margins all alternative baselines and achieves state-of-the-art performance with a 71.4\% and 87.2\% Top-1 accuracy respectively. Code is available at https://github.com/xyzforever/BEVT.

1. Introduction

Transformers \cite{vaswani2017attention, liu2019swin} have become the dominant network structures in the natural language processing (NLP) field and have demonstrated tremendous success in different NLP tasks. Recently, the pioneering work ViT \cite{dosovitskiy2020image} tokenizes one image into a series of patch-based tokens and applies the transformer architecture for image recognition. Many approaches \cite{chen2021generative, he2021masked, tan2022efficient, yang2022make} further demonstrate the power of transformers as generic vision backbones and achieve impressive performance on various vision tasks. Beyond image tasks, there are also a few studies showing the promise of transformers for video understanding \cite{carreira2017quo, xia2021video}.

The key to the success of Transformers in NLP is BERT pretraining \cite{devlin2018bert, liu2019roberta, tan2019lxmert}, one of the most successful pretraining tasks, which predicts masked tokens in corrupted texts. This motivates a few recent studies to explore the BERT-style pretraining for image representation learning by recovering raw pixels \cite{wang2021video} or latent codes \cite{caron2021infomax, he2022masked} of masked image patches. However, how to leverage such a strategy for video understanding has never been explored.

In this paper, we study BERT pretraining of video transformers. Unlike static images, videos depict how objects move and interact over time. Such dynamic nature brings additional difficulty for representation learning. It is often found that learning representations from scratch on videos is computationally expensive and requires extremely large-scale datasets with millions of samples \cite{caron2019unsupervised}, if not hun-
dreds of millions of samples [1]. Instead of training from scratch, a few methods demonstrate that self-supervised models pretrained on image datasets benefit video recognition under both supervised [2, 37] and unsupervised settings [5]. These approaches simply leverage pretrained models as better initializations to learn spatial-temporal features in videos. While widely used and sometimes effective, the spatial context relationships learned from the image pretraining phase are likely to be drastically modified during video feature learning.

We argue that spatial priors encoded in pretrained self-supervised models should be explicitly preserved when performing video representation learning. The intuition behind is that there are large inter-class variations among different videos and their dependencies on what discriminative information to use (i.e., spatial and temporal clues) to make correct predictions differ. For instance, for actions like “applying lipstick”, spatial knowledge is generally sufficient, as evidenced by the fact simply using 2D features offers decent results on datasets like Kinetics [9]. On the other hand, temporal dynamics are crucial for differentiating actions between two fine-grained diving sequences [33]. This highlights the importance of considering the differences among video samples during feature learning.

In light of this, we introduce BEVT, which decouples video representation learning into spatial representation learning and temporal dynamics learning. More specifically, BEVT builds upon the Video Swin Transformer [37] due to their computationally efficient architectures *, and is trained with a BERT-style objective to fully unleash the power of transformers for representation learning. BEVT contains an image stream for spatial modeling and a video stream for temporal modeling, interacting with each other for video modeling. In particular, the image stream, operating on RGB images, learns spatial priors first in an unsupervised fashion by predicting masked image patches in the form of latent codes derived from a pretrained VQ-VAE [3]. It is then used to initialize the attention weight matrices of the video stream, whose inputs are sampled video clips, so as to save computation for video transformers. The video stream, on the other hand, learns temporal dynamics through predicting masked 3D tubes represented by latent codes. The two streams, taking image and video pairs as inputs, are then jointly trained on video data through a weight sharing strategy. Such a design not only maintains spatial knowledge learned from image datasets to ensure decent results for static video samples but also learns temporal information to guarantee correct predictions for samples that contain dynamic movements. Finally, BEVT is finetuned on targeted datasets for downstream evaluation.

We conduct extensive experiments on three challenging video datasets, i.e., Kinetics-400 (K400) [9], Something-Something-v2 (SSV2) [26], and Diving-48 (DIVING-48) [33]. On K400, BEVT offers 81.1% Top-1 accuracy, which is better than the strong supervised baseline 80.6% [37]. On SSV2 and DIVING48, BEVT achieves 71.4% and 87.2% Top-1 accuracy outperforming state-of-the-art methods [2, 5, 23, 37] by clear margins. To further analyze the performance difference among these three datasets, we further provide the temporal dependency analysis and demonstrate that videos in K400 mainly rely on spatial clues for correct predictions while videos from SSV2 and DIVING48 require more temporal information.

Our main contributions are summarized as follows: (1) We explore the BERT-style training objective to fully unleash the power of transformers to learn discriminative video representations; (2) We introduce a novel two-stream network that decouples spatial representation learning and temporal dynamics learning; (3) We demonstrate different video samples have different preferences towards spatial and temporal clues; (4) We conduct extensive experiments on three challenging video benchmarks and achieve comparable or better results with state-of-the-art methods.

2. Related Work

Video understanding with CNNs. There is a plethora of work on video understanding with CNNs, most of which focus on learning spatial-temporal features [9, 22, 23, 34, 50, 53, 57, 61, 62]. These approaches can be divided into two categories: (1) temporal aggregation and (2) 3D CNNs. In particular, temporal aggregation methods typically extract image features/scores frame-by-frame and then combine frame-level information to achieve video-level predictions through recurrent networks [15, 66] or average pooling [47, 57]. On the other hand, 3D CNNs extend 2D convolutions into the time domain by using 3D convolutions on stacked RGB frames for the joint-modeling of spatial-temporal relationships [9, 22, 23, 50, 53]. 3D CNNs are generally computationally expensive, and this motivates a line of research on efficient video recognition [13, 22, 34, 51, 53, 72, 74]. Instead of using CNNs, we explore transformers for video understanding due to their strong results on image recognition tasks.

Vision transformers. Motivated by the impressive performance of transformers in a wide range of NLP tasks, there is a growing interest in using transformers for computer vision tasks [19, 20, 28, 36, 38, 43, 49, 56, 73]. More specifically, ViT [19] generalizes transformers to the image domain by splitting images into patches which are further embedded with a linear layer as inputs to a Transformer. While demonstrating great potential in image recognition tasks, ViT relies on pretraining on substantially large-scale datasets and the training process is computationally expensive. To mit-
igate these issues, extensive studies have been introduced. For example, DeiT [49] uses a distillation loss to speed up the training process. CSwin Transformer [17] proposes multi-head grouping and performs attention within cross-shaped windows. Mobile-Former [12] further introduces lightweight transformers. There are also a few very recent studies that extend image transformers for video understanding [2, 20, 37, 68]. Fan et al. use a multi-scale design to generate spatial-temporal tokens in different sizes for action recognition [20]. Liu et al. extend Swin Transformers into the video domain [37]. In contrast, we study BERT pretraining of video transformer in a self-supervised way, which is orthogonal to transformer network design.

Self-supervised representation learning. At the core of many computer vision tasks is how to learn discriminative features for targeted datasets. Since collecting labeled datasets is labor-intensive and costly, there is an ever-increasing trend in learning representations in a self-supervised manner [6, 7, 16, 27, 32, 67]. The main idea is to design surrogate tasks including inpainting [44], colorization [70, 71], jigsaw predictions [42], rotation predictions [25], etc., as a form of supervisory signals in lieu of manual labels. More recently, contrastive learning has been a popular paradigm for feature learning by forcing images to be closer to their augmented copies than other samples [10, 30, 40, 63, 65]. In contrast to these approaches using CNNs as backbones, there are a few very recent studies leveraging contrastive learning [8, 11] for transformers.

BERT pretraining. In contrast to contrastive learning widely used in vision, BERT pretraining [14] is extremely popular and extensively studied [4, 35] in NLP. As an effort that unifies vision and NLP under the same BERT pretraining framework, the recent work BEiT [3] and ICT [55] utilizes the masked image modeling task to do BERT pretraining of image transformers and achieves great success for different tasks. And one concurrent work PeCo [18] further proposes a perceptual codebook to improve the performance. Another concurrent work [29] extends it from recovering patch tokens to raw pixels. In this paper, we study BERT pretraining for video transformers as an orthogonal unifying effort. Different from BERT pretraining of image transformers and the concurrent effort [48], we decouple video pretraining into spatial representation learning and temporal dynamics learning so as to accommodate the varying need of distinct salient clues for different videos.

3. Method

The goal of BEVT is to learn video representations effectively for both relatively static videos and dynamic videos in a self-supervised manner. Here, “relatively static videos” mean the videos only requiring discriminative spatial representation for recognition, while “dynamic videos” mean that videos that also require temporal dynamics for recognition. Besides the effectiveness, another key problem to consider in video pretraining is efficiency. Compared to image pretraining, video pretraining is more computationally expensive, thus making pretraining on large-scale video data from scratch inefficient or even inapplicable without massive computational resources.

To this end, BEVT decouples the video pretraining into spatial representation learning and temporal dynamics learning. And the spatial representation learning is only conducted on image data, while the temporal dynamics learning is conducted on video data. To implement this idea, our BEVT contains two streams, operating on images and videos, respectively. In the following, we introduce different components of our framework. Figure 2 gives an overview of our framework.

Image and video patches. For the video stream, given a video clip $X_{vid} \in \mathbb{R}^{T \times H \times W \times 3}$ with $T$ frames, we follow VideoSwin [37] and convert it into $\frac{T}{4} \times \frac{H}{4} \times \frac{W}{4}$ 3D patches, each with a size of $2 \times 4 \times 4 \times 3$; each 3D patch contains a 96-dimensional features. For the image stream, given an input image $X_{img} \in \mathbb{R}^{H \times W \times 3}$, we consider each patch with a size of $4 \times 4 \times 3$ as a token and set the feature dimension of each token as 48. We then project each token to a token embedding vector of dimension $C$ by a linear embedding layer. Then the sequence of token embedding is input into the following transformer architectures.

Masked image and video tokens. Motivated by the great success of BERT in NLP tasks, BEVT is optimized to simultaneously perform masked image modeling (MIM) and masked video modeling (MVM) by predicting “corrupted” image and video tokens, respectively. The MIM is designed to capture spatial priors while the MVM is used to capture temporal dynamics in videos. In particular, for the image stream, since input images are divided into non-overlapping patches, we randomly mask several patches and the image stream is trained to recover them as in [3]. More specifically, the embedded feature of each masked patch is replaced by a learnable mask token embedding. For the video stream, we randomly mask 3D tokens and train the video stream to predict those masked tokens. The set of masked image and video tokens and the remaining patch features are sent to encoders, as will be introduced below.

Mask strategy. For masked image modeling, following [3], we use blockwise masking instead of randomly selecting each masked patch. When generating masked positions for an image, we mask a block of patches each time and set the minimum number of patches for each block. The position, the aspect ratio and the size of each block are randomly selected under a preset range. We repeat masking blocks until the ratio of masked patches exceeds the preset lower bound. For masked video modeling, we employ
a tube masking strategy that is a straightforward extension of blockwise masking. Given an input video clip of length $T$, we first randomly choose the number of masked frames (tube length) $l$ and the start frame $t$. Then we employ blockwise masking to generate a 2D mask, and apply this 2D mask to each frame from $t$ to $t + l$. In other words, for each masked frame, the set of masked positions is the same and the shape of the whole 3D mask is a tube. The range of the masked tube length is $[0.5T, T]$ and the masking ratio of each masked frame is 0.5.

**BEVT encoders.** BEVT contains two encoders, one for the image stream and one for the video stream. Both encoders are instantiated with the Video Swin Transformer [37] due to its strong performance with a moderate computational cost. Note that in contrast to [37] that performs fully-supervised training, we use the Video Swin Transformer as our backbone for self-supervised learning. In particular, Video Swin Transformer [37] follows the design of Swin Transformer [36] and is a hierarchical architecture consisting of four stages. Between every two stages, spatial downsampling is performed by patch merging layers, which concatenates the features of each group of $2 \times 2$ spatially neighboring patches. After downsampling, a linear layer maps the features of each concatenated token to half of their dimension. A series of Swin attention blocks comes after to apply feature transformation.

Given a sequence of tokens as inputs, the video encoder outputs a feature map with the size of $\frac{H}{32} \times \frac{W}{32} \times 8C$. Since Video Swin Transformer only performs temporal downsampling in the beginning linear embedding layer, it degrades to a 2D architecture when the temporal dimension of the input is 1. As a result, for the image encoder, the output feature map has a size of $\frac{H}{32} \times \frac{W}{32} \times 8C$.

**Tokenizer.** Following [3], we use the visual tokens generated by a pretrained image VQ-VAE [46] as the groundtruth tokens and our pretraining task is to predict the tokens for masked patches. The pretrained VQ-VAE tokenizer maps image patch into discrete tokens $z$ by searching the closest latent codes in its pre-learnt visual codebook. Given an input image $X_{\text{img}} \in \mathbb{R}^{H \times W \times 3}$, it will be tokenized into the visual token map $Z_{\text{img}} \in V^{\frac{H}{32} \times \frac{W}{32}}$. Similarly for an input video $X_{\text{vid}} \in \mathbb{R}^{T \times H \times W \times 3}$, it is tokenized into visual token map $Z_{\text{vid}} \in V^{T \times \frac{H}{32} \times \frac{W}{32}}$. Note that, considering the pretrained VQ-VAE only downsamples 8 $\times$ 8 patch into one token, we downsample the input images/frames by 1/2 before feeding into the tokenizer so that the output token map has the spatial resolution of $\frac{H}{16} \times \frac{W}{16}$.

**BEVT decoders.** To learn meaningful representations by predicting the tokens for the masked image and video patches, BEVT has an image decoder and a video decoder as the auxiliary prediction heads, which will be discarded in finetuning stage. Existing modern vision transformers including Swin Transformer follow the hierarchical design and downsample the input into decreased spatial/temporal resolutions. Taking the VideoSwin of video stream shown in Figure 3 as an example, it consists of four stages, and the feature maps $F_4$ in the last stage have the dimension of $\frac{T}{2} \times \frac{H}{32} \times \frac{W}{32}$. In order to match the dimension of feature maps to the number of groundtruth visual tokens, we design a lightweight decoder for the video stream in BEVT. As shown in Figure 3, it first spatially upsamples the stage-4 feature $F_4$ by using a transposed convolutional layer, and then concatenate the upsampled stage-4 feature $\hat{F}_4$ with
Figure 3. BEVT encoder and decoder for masked video modeling.

Stage 1 feature $F_3$ together and fuse them with a simple linear layer. Finally, the fused feature $F$ will be temporally upsampled with another transposed convolution layer to obtain $\hat{F}$. To predict the token for each position $(t, i, j)$, a simple softmax based classifier is applied upon $\hat{F}$.

$$p_{t,i,j} = \text{softmax}(W f_{t,i,j} + b)$$  (1)

where $f_{t,i,j}$ is the feature vector of the output feature map $\hat{F}$ at position $(t, i, j)$, $p_{t,i,j}$ denotes the corresponding probability vector, $W$ and $b$ are the weight and the bias of a linear layer. For the decoder in the image stream, it follows a similar design, and the only difference is without the temporally upsampling part.

Training objectives. Denote the positions of masked patches in the input images and videos as $M_I$ and $M_V$, the objective of masked image modeling is to maximize the log-likelihood of the groundtruth token $z_{t,i,j}$ for each mask position $(i, j)$.

$$L_{MIM} = -\frac{1}{|M_I|} \sum_{(i,j) \in M_I} \log p_{z_{t,i,j}}^{i,j}$$  (2)

where the superscript of $p$ denotes indexing the probability value of one specific position. Similarly, the target of masked video modeling can be denoted as:

$$L_{MV M} = -\frac{1}{|M_V|} \sum_{(t,i,j) \in M_V} \log p_{z_{t,i,j}}^{t,i,j}$$  (3)

The objective of two-stream joint training is a simple combination of two objectives:

$$L = L_{MIM} + \lambda L_{MV M}$$  (4)

where $\lambda$ is the hyper-parameter that balances the weights of the image stream and the video stream.

Training strategies. Following our decoupled design, we first train the image stream on ImageNet with the masked image modeling task to learn discriminative spatial representation. The resulting model is then used to initialize the video stream, and both streams are jointly trained by optimizing Equation 4 such that the objective $L_{MIM}$ preserves spatial information while $L_{MV M}$ learns to capture temporal dynamics in videos. Such a strategy not only makes BEVT much more efficient than pretraining video transformers on large-scale video data from scratch, but also satisfies the need of learning different discriminative clues for different types of video samples.

Weight sharing between streams. When jointly training the image and video stream, instead of learning two sets of model weights for the two streams independently, we design a weight sharing strategy so that they can share model weights for the encoder except some image/video specific parts. This is motivated by the good property of transformer networks, i.e., most operators (including multi-head attention and FFN) are oriented to tokens but not specific input types. Taking the Video Swin transformer as an example, we use the following strategies for weight sharing: (1) We use independent 2D patch partitioning layers instead of 3D patch partitioning, and add a linear embedding layer in the first stage for projecting image tokens to the same dimension as the original 3D video patch embedding; (2) We adapt 3D shifted local window to the 2D scenario. This is fulfilled by reusing the submatrix of the original 3D relative positional embedding where the relative temporal distance is 0 as the 2D relative positional embedding. With such a design, the image stream and the video stream can help each other by optimizing one “mostly-unified” encoder.

Fintuning and inference. Once pretrained, BEVT provides decent video representations that can be transferred for downstream tasks. On targeted datasets, we simply use the 3D patch embedding layers and the video encoder, to which a few task-specific layers (e.g., classification head for video recognition) are appended, for fintuning. The resulting model can then be readily used for inference.

4. Experiments
4.1. Experimental Setup

Datasets and evaluation metrics. We evaluate our method on three representative video recognition datasets: Kinetics-400 (K400) [9], Something-Something-v2 (SSv2) [26], and Diving-48 (DIVING48) [33]. K400 contains videos clips from YouTube with an average duration of 10 seconds and the videos are manually labeled into 400 categories. Following [23], we use ~240K videos for training and ~20K videos for testing. SSv2 is also a large-scale video dataset that contains ~160K videos for training and ~20K videos for testing. The videos in SSv2 are labeled into 174 classes the average duration is 4 seconds. DIVING48 contains ~17K fine-grained diving sequences, which are fur-
ther split into a training set with around $\sim15K$ clips and a testing set with $\sim2K$ clips. Compared to K400, recognizing videos in SSV2 and DIVING48 requires more temporal information, as will be introduced below. Following official instructions, we report Top-1 accuracy on all three datasets. And the default resolution $224 \times 224$ is used.

**Implementation Details.** We use Video Swin-Base for experiments throughout the paper unless mentioned otherwise. For pretraining the image stream BEVT-I alone, we train for 800 epochs on ImageNet-1K with a batch size of 2048. For pretraining the video stream BEVT-V alone or the two-stream BEVT, we train for 150 epochs on K400 with a batch size of 256 and the clip length $T$ is 16. For two stream pretraining, IMAGE-Net images are used to train the image stream with a batch size of 2048, and the loss weight $\lambda$ is simply set to 1. We use the DALL-E tokenizer [46] unless explicitly stated. The image stream pretraining takes about 4 days on 32 V100 GPUs. The two-stream pretraining of 150 epochs takes about 4 days on 32 V100 GPUs.

### 4.2. Main Results

**Effectiveness of BEVT for video transformer pretraining.** To demonstrate the effectiveness of BEVT, we compare it with the four image stream pretraining baselines: (1) *Image Sup:* pretraining the image Swin Transformer on ImageNet-1K in a supervised way. Similar strategies are commonly used in existing video transformer papers [2, 5, 37]. (2) *Image CL:* pretraining the image Swin Transformer on Imagenet-1K with a self-supervised contrastive learning method [64]. (3) *BEVT-I:* pretraining the image Swin Transformer only with the image stream, which is similar to BeIT [3]. (4) *BEVT-V:* pretraining the video Swin Transformer only with the video stream. The pretrained weights from *Image Sup, Image CL* and *BEVT-I* used as the initialization of the Video Swin Transformer for finetuning. For the video stream, we design two baselines by conducting BERT pretraining on K400 and HowTo100M [39] from scratch, i.e., *BEVT-V*, which is our framework without the decoupled design. As emphasized before, because video pretraining is more computationally expensive than image pretraining, pretraining on HowTo100M dataset with many epochs is not applicable. For fair comparisons, we also use 32 V100 GPUs to pretrain HowTo100M about 8 days (about 2 epochs).

The comparison results are summarized in Table 1. We observe that: (1) BEVT outperforms the *Image Sup* baseline by clear margins (4.3% and 2.7%) on SSV2 and DIVING48, respectively. This not only suggests that learning representations with BEVT using a BERT-style training objective is promising without the need for manual labels, but also shows that only image-based pretraining is not enough for these two datasets. On K400, the performance of BEVT is on par with the *Image Sup* baseline. (2) We also see that BEVT offers comparable or better results compared to *Image CL* on these three datasets. (3) Compared to BEVT-I, BEVT is better on SSV2 and DIVING48 by 1.4% and 5.5% respectively, highlighting the gains brought by the video stream. Similarly, BEVT obtains similar results as BEVT-I on K400. (4) Compared to BEVT, BEVT-V pretraining on K400 or HowTo100M from scratch under the similar computation budget achieves much worse results. We hypothesize it may be because the data diversity of K400 is not as good as ImageNet. And for HowTo100M, pretraining much more epochs may help learn better video representation, but it too costly. This also further justifies the decoupled design in our BEVT.

**Deeper dataset analysis.** To further understand the performance variations of BEVT among these datasets, we perform a temporal dependency study to investigate the amount of temporal information required for correct predictions. Specifically, we use the following two testing strategies: (1) Single-frame, where we randomly sample a frame and replace all other frames with this one, leading to a static video; (2) Random-Shuffling, where a random shuffling is performed along the temporal axis. The results are summarized in Table 2. We observe that both strategies have a relatively small impact on K400 compared to SSV2 and DIVING48 where there is a 60% and 70% performance drop when using the Single-frame strategy. This suggests most videos in K400 can be recognized by discriminative spatial clues whereas temporal dynamics is particularly important for SSV2 and DIVING48.

![Table 1. Comparison of different pretraining methods. Video Swin-Base is used here.](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>Pretrain</th>
<th>SSV2</th>
<th>DIVING48</th>
<th>K400</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Sup</td>
<td>IN-1K</td>
<td>66.3</td>
<td>84.0</td>
<td>80.6</td>
</tr>
<tr>
<td>Image CL</td>
<td>IN-1K</td>
<td>67.1</td>
<td>85.5</td>
<td>80.9</td>
</tr>
<tr>
<td>BEVT-I</td>
<td>IN-1K</td>
<td>69.2</td>
<td>81.2</td>
<td>80.5</td>
</tr>
<tr>
<td>BEVT-V</td>
<td>K400</td>
<td>67.1</td>
<td>83.7</td>
<td>76.2</td>
</tr>
<tr>
<td>BEVT-V</td>
<td>HowTo100M</td>
<td>64.2</td>
<td>82.3</td>
<td>75.1</td>
</tr>
<tr>
<td>BEVT</td>
<td>IN-1K+K400</td>
<td>70.6</td>
<td>86.7</td>
<td>80.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Normal</th>
<th>Single-frame</th>
<th>Random-Shuffling</th>
</tr>
</thead>
<tbody>
<tr>
<td>K400</td>
<td>80.6</td>
<td>65.3</td>
<td>77.8</td>
</tr>
<tr>
<td>SSV2</td>
<td>66.3</td>
<td>6.3</td>
<td>19.0</td>
</tr>
<tr>
<td>DIVING48</td>
<td>84.0</td>
<td>13.8</td>
<td>50.4</td>
</tr>
</tbody>
</table>
decent performance. Additional video modeling brings little effect to the overall performance; (2) The use of the video stream in BEVT is crucial to learn necessary temporal information for datasets like SSv2 and DIVING48. This confirms our hypothesis that different videos rely on different discriminative clues for accurate predictions due to the large intra-class and inter-class variations among videos.

**Comparisons with State-of-the-art methods.** We compare BEVT with alternative methods on SSv2, DIVING48 and K400. On SSv2 and DIVING48, we see from Table 3 and Table 4 that our approach achieves the best performance by clear margins when compared to existing SOTA methods, including supervised models. It is worth mentioning that, on SSv2, a common practice to achieve better results is to perform two rounds of pretraining—a model is pretrained on both ImageNet and K400 in a fully supervised fashion before finetuning the model on SSv2. Instead, we pretrain on ImageNet and K400 without using any manual labels, yet our performance is still better. On K400, we see from Table 5 that BEVT achieves competitive results with SOTA methods using similar or less computation measured by GFLOPs.

### 4.3. Ablation Study

**Importance of image stream pretraining.** In our BEVT, we first conduct the image stream pretraining only on the large-scale image data to efficiently learn spatial representations, then use it as the initialization for joint pretraining. To show its importance, we provide some ablation results in Table 6, where the column “Init” means whether to use the image stream pretrained weights as initialization or not. We have some interesting findings: (1) Using the image stream pretrained weights as initialization can benefit both pure video stream pretraining (i.e., “BEVT-V”) and the following joint pretraining of image stream and video stream (i.e., “BEVT”). (2) Even with the initializations, jointly training the image stream with the video stream is still necessary and can bring desirable performance gain.

**Image data in joint training.** By default, when jointly learning spatial and temporal representations, the image stream in BEVT continues to use the ImageNet images as training images. In this ablation, we also experiment with a variant that uses K400 frames for the image stream. The results are shown in Table 7. We see that images from ImageNet are slightly better than those from K400, i.e., less than 0.3% on all three datasets. This suggests that the image stream, designed to preserve spatial knowledge, is not very sensitive to data domains.

**Different Pretrained Tokenizer.** We also experiment with the PeCo tokenizer [18] instead of the DALL-E tokenizer [46] during pretraining.
Method Init SSv2 DIVING48 K400
BEVT-I - 69.2 81.2 80.5
BEVT-V × 67.1 83.7 76.2
BEVT-V ✓ 70.0 85.2 79.6
BEVT × 67.9 85.1 78.5
BEVT ✓ 70.6 86.7 80.6

Table 6. Ablation study to show the importance of image stream pretraining. Init means models are initialized from image transformers pretrained with the image stream on ImageNet-1K.

Method Video Image SSv2 Diving48 K400
BEVT-V K400 - 70.0 85.2 79.6
BEVT K400 K400 70.3 86.6 80.5
BEVT K400 IN-1K 70.6 86.7 80.6

Table 7. Ablation study on the image data of joint pretraining. Models are initialized from image transformers pretrained with the image stream on ImageNet-1K.

Method Pretrain SSv2 DIVING48 K400
Image Sup IN-1K 57.6 77.4 75.4
BEVT-I IN-1K 61.8 77.2 76.5
BEVT IN-1K+K400 65.7 82.3 77.1

Table 8. Comparison of different pretraining methods with TimeSformer [5]. PeCo tokenizer is used here.

dom-Frame: which samples random frames and masks them with the same 2D mask generated by blockwise masking. The temporal positions of masked frames may not be consecutive. The results are summarized in Table 9. We have several observations: (1) Masking tubes offers the better results compared to other masking methods like Random-3D and Random-Frame. (2) Setting too small tube temporal length (e.g., [0.25T, 0.75T]) or too large temporal length (e.g., T) will both incur inferior results on SSv2. We guess it is because the former setting will make the masked video modeling too easy while the later will degrade to masked image model to some extent. (3) Applying different block masks for different frames (“Frame-Diff”) is also not good, which possibly shares the similar reason as the small temporal length, i.e., making masked video modeling too easy because information can be easily borrowed from adjacent/short-term frames.

5. Conclusion and Discussion

Benefited from the BERT pretraining strategy, transformers have dominated various NLP tasks. Recent success of transformers in computer vision motivates the use of BERT pretraining for feature learning. We study how to explore BERT pretraining for video transformers. We introduced BEVT that learns both discriminative spatial and temporal representations using a carefully designed decoupling strategy, which is not only efficient but also effective. BEVT achieved SOTA performance on three video recognition datasets. Although our decoupled design is efficient, BEVT is still computationally intensive and exploring training with constrained resources is left as future work.

Acknowledgement This project was supported by National Key R&D Program of China (No. 2021ZD0112805). Y.-G. Jiang was sponsored in part by “Shuguang Program” supported by Shanghai Education Development Foundation and Shanghai Municipal Education Commission (No. 20SG01). Z. Wu was supported by NSFC under Grant No. 62102092.
References


[28] Dongyoon Han, Sangdoo Yun, Byeongho Heo, and Youngjoon Yoo. Rethinking channel dimensions for efficient model design. In CVPR, 2021.

[29] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. arXiv preprint arXiv:2111.06377, 2021.


