Leaping Into Memories: Space-Time Deep Feature Synthesis

Alexandros Stergiou  Nikos Deligiannis
Vrije Universiteit Brussel, Belgium & imec, Belgium
<first>.<last>@vub.be

Abstract

The success of deep learning models has led to their adaptation and adoption by prominent video understanding methods. The majority of these approaches encode features in a joint space-time modality for which the inner workings and learned representations are difficult to visually interpret. We propose LEArned Preconscious Synthesis (LEAPS), an architecture-independent method for synthesizing videos from the internal spatiotemporal representations of models. Using a stimulus video and a target class, we prime a fixed space-time model and iteratively optimize a video initialized with random noise. Additional regularizers are used to improve the feature diversity of the synthesized videos alongside the cross-frame temporal coherence of motions. We quantitatively and qualitatively evaluate the applicability of LEAPS by inverting a range of spatiotemporal convolutional and attention-based architectures trained on Kinetics-400, which to the best of our knowledge has not been previously accomplished. 1

1. Introduction

Inverting deep networks’ learned internal representations has been a difficult task to achieve. The adaptation and deployment of CNNs and more recently Transformers, to a variety of vision tasks has led to significant breakthroughs. The field of video action recognition has experienced drastic growth in recent years through the convergence of models with increased complexities and capacities [2, 4, 32, 63] as well as large accuracy improvements [11, 12, 27, 26, 29]. Despite great progress in their applicability, there is still a large gap in the interpretability of video models. As these models compositionally encode space and time modalities of videos over large feature spaces and require a lot of parameters, conceptualizing their internal representations remains challenging. In this paper, we propose a method to invert learned features of video models associated with specific actions by optimizing a parameterized input to synthesize conceptually and visually coherent representations.

In cognitive science, stages of consciousness include the conscious, the unconscious, and the preconscious. In contrast to the conscious and unconscious, the preconscious is

Figure 1: Illustration of our proposed LEAPS model inversion. Given a video input $x^*$ initialized with noise, we use a stimulus video $v$ of target class $y$, to prime video model $\Phi(\cdot)$. We iteratively optimize $x^*$ to synthesize and visually represent the internal spatiotemporal features of $\Phi(\cdot)$. To impose diversity over the synthesized videos, feature statistics from a domain-specific verifier network $S(\cdot)$ are distilled to regularize $x^*$. To preserve the continuity of motions across frames, $x^*$ is temporally regularized at each training iteration. Although the illustration visualizes internal representations as $C \times T \times H \times W$ volumes, they can also be flattened patches $C P^3 \times \frac{T H W}{P^3}$ of $P^3$ resolution as in Vision Transformers.
responsible for learned information currently outside conscious awareness to remain readily available [18]. One way of accessing learned preconscious information and making it part of conscious awareness is through priming [38]. Priming uses a stimulus to activate related learned concepts in memory and make them easily and readily accessible [33, 34]; e.g., one can remember their bedroom if primed with a picture of a bed.

Motivated by visual priming in cognitive science, we demonstrate that learned representations of video models can become accessible through model priming. By using a video stimulus and a target action class, we synthesize the dominant learned concepts corresponding to actions. In turn, the visual features of the synthesized videos provide a conceptual view of the models’ learned internal representations. We term these features as the learned preconscious of the video model associated with a specific action.

We introduce LEA(rned) Preconscious Synthesis (LEAPS), illustrated in Figure 1, a spatiotemporal model inversion method that synthesizes interpretable videos by minimizing a classification and priming loss without prior knowledge of the training data. LEAPS uses a video of a target action class as stimulus, to prime a fixed spatiotemporal model. We additionally include two regularization terms. The first term enforces motion coherence across frames by constraining their representations at each update. The second enhances the diversity of the synthesized videos by using a domain-specific verifier network that exploits disagreements between feature statistics similar to [64]. Through the same architecture-independent approach, we show that LEAPS can invert the spatiotemporal features of 3D-CNNs and spatiotemporal Transformers.

Our main contributions are as follows. First, we introduce LEAPS, a general approach for inverting video models, which to the best of our knowledge, is the first attempt to create videos of conceptual representations from jointly encoded space-time features. Second, we use LEAPS on multiple convolution and attention models and compare it to prior image-based inversion methods that we extend to video. Finally, we show that LEAPS can invert both video CNNs and transformers, with the same architecture-independent method without any modifications.

2. Related Work

Approaches for visualizing and interpreting deep models can be divided into three groups which we detail below.

Attribution-based visualizations. These methods have been used to visualize feature contributions over an input. Attribution methods for images have primarily been based on back-propagating activations of classes [8, 48, 61], or individual neurons [3, 5, 49, 52] to localize regions in the input that are informative for a given class or feature. Such approaches include Integrated gradients (IG) [56] which produce pixel-wise attributions from integrating computed backprop gradients. Subsequent works have also included gradient smoothing [51], gradient accumulation in saturated regions [35], and adaptation of the gradient path [23]. Another set of approaches that rely on attribution-based visualizations use perturbations of the input [17, 16, 44]. Given their straightforward applicability, image attribution methods have also been extended to video. The majority of works; e.g., Saliency Tubes [55, 54], STEP [28], video OSA [59], and BOREx [24], have focused on the localization of spatiotemporal salient regions by extending Grad-CAM [48], Extremal Perturbations (EP) [16], Occlusion Sensitivity Analysis (OSA) [66], and Gaussian processes regression (GPR) [6, 37] respectively. In contrast to these approaches that localize salient class features, we offer a visual feature synthesis method to conceptualize the learned representations of video models through priming.

Input synthesis. Network-centric approaches invert models to either visualize particular classes [41, 65, 64] or features [43, 53]. One of the main methods employed for visualizing internal network features is Gradient Ascent (GA) [10], which optimizes the input by increasing the activation of a specific neuron. Activation Maximization (AM) [50] later adapted GA to visualize CNN features. Following works have been built on top of AM by including additional regularizers; e.g., total variation [31], blurring [60], and gradient masking [41]. One of the most popular extensions of AM has been DeepDream [1] which optimizes the input to yield high responses for a chosen class while keeping internal representations constraint-free. The produced images include repetitions of recognizable concepts without representing a coherent whole. In addition to AM, model inversion [9, 19, 30, 64, 66] includes the task of maximizing the classification score of the synthesized image instead of maximizing class activations. The only extension of input synthesis approaches to videos has been introduced by Feichtenhofer et al. [13] in which AM is used to create visual representations of class features from two-stream models [14, 62], trained on RGB frames (spatial stream) and optical flow (temporal stream). In this paper, we instead propose a model inversion method for inverting video models concurrently encoding space and time modalities.

Visual feature generation. These methods utilize activation maximization by including an additional generator network [39, 40, 42], with the cost of requiring access to the training data. Huang et al. [21] adapted feature generation on video data by modeling the temporal signal with a temporal generator network. Despite the high fidelity of the produced results, these approaches are less suitable for interpreting internal model behaviors, as they do not solely depend on the model under inspection. Instead, they are primarily influenced by the training data used as well as the capacity and complexity of the generator model trained.
3. Learned Preconsciously Synthesis

In this section, we overview our LEAPS method, shown in Figure 1. We start by formally introducing model priming for video models in Section 3.1. A stimulus video of a target class is used to initially prime the network. The training process uses the primed representations to update a randomly initialized input alongside two regularizers. The first regularizer is used to enforce temporal coherence, explained in Section 3.2. The second is used to improve synthesized feature diversity, overviewed in Section 3.3. We present the final aggregated function in Section 3.4.

3.1. Model Priming

Priming deep models is influenced by cognitive science [38] in which a stimulus is used to recall prior knowledge. We extend this approach to visualizing the learned preconsciously of deep video models associated with a specific action class \( y \). Given a video \( v \) of size \( C \times T \times H \times W \), with \( C \) channels, \( T \) frames, \( H \) height, and \( W \) width, as a visual cue for action class \( y \), and a randomly initialized input \( x^* \) of \( C \times T \times H \times W \) size to optimize. We define a priming loss \( L \text{prim}(x^*, v) \) between the internal representations \( z^l(\cdot) \) of the optimized input \( x^* \) and stimulus \( v \) across \( l \in \Lambda = \{1, ..., L\} \) layers:

\[
L \text{prim}(x^*, v) = \frac{1}{L} \sum_{l \in \Lambda} \lambda_l JSV \left( \mu(z^l(x^*)), \mu(z^l(v)) \right) \tag{1}
\]

where \( \mu(z^l(\cdot)) \) is the \( C \)-length spatiotemporal mean vector of representations \( z^l(\cdot) \). \( JVS(\cdot) \) denotes the Jaccard vector similarity [15]. To integrate a degree of freedom and avoid hard constraints on the internal representations of \( x^* \), we define \( 0 < \lambda_l \leq 1 \) as priming weight for layer \( l \). An overview of updating \( x^* \) by priming appears in Figure 1.

Due to the vastness of the feature space when optimizing (1), we include two additional regularization terms to constrain the input. Specifically, we apply a temporal coherence regularization \( R_{coh} \) and a feature diversity regularization \( R_{feat} \), which we detail below.

3.2. Temporal Coherence Regularization

For the first regularizer, we aim to enforce similarity between representations of consecutive frames in order to enable consistent feature transitions in the synthesized video. Therefore, we include a coherence regularizer \( R_{coh} \) formulated based on the temporal coherence loss from [36]. Given two spatiotemporal representations \( z^l(x^*)_{t_1} \) and \( z^l(x^*)_{t_2} \) at layer \( L \), for temporal locations \( t_1 \) and \( t_2 \), we use their \( l_1 \) norm to enforce similarity if \( t_1 \) and \( t_2 \) are consecutive in the video. In non-consecutive cases, the divergence between \( z^l(x^*)_{t_1} \) and \( z^l(x^*)_{t_2} \) should increase.

\[
R_{coh}(x^*) = \begin{cases} 
0 & \text{if consecutive} \\
\max(0, \delta - \|z^l(x^*)_{t_1} - z^l(x^*)_{t_2}\|_1) & \text{elsewise}
\end{cases} \tag{2}
\]

where \( \delta \) is a margin hyperparameter. Temporal coherence is enforced at layer \( L \) as in [36]. Although \( R_{coh} \) can be applied to any layer \( l \in \Lambda \), in practice, minimizing (2) for all layers enforces a very strong regularization, producing synthesized videos with minimal to no cross-frame variations as shown in Figure 2. We note that for Transformers using patches of \( P^3 \) resolution, we first reshape \( z^l(x^*) \) from \( C' \times P^3 \times THW^3 \) to \( C' \times T' \times H'W' \) before calculating (2).

3.3. Feature Diversity Regularization

Our second regularization term \( R_{feat} \) is responsible for improving the diversity of features generated by model priming. Although priming provides a strong signal based on which the input can be updated, the diversity of features is limited compared to observing multiple instances. Thus, class features varying from those in the stimulus, or features not present in the stimulus, may not be explored during optimization. In order to enhance the search space we introduce an additional domain-specific verifier network \( S(\cdot) \). Our goal is to use high- and low-level feature distribution statistics as proposed in [64] incorporating the verifier’s prior knowledge during optimization.

Based on input \( x^* \), we run inference on the verifier \( S(x^*) \) to obtain representations \( a^k(x^*) \) across each verifier layer \( k \in K \). The feature statistics are then obtained by the \( C \)-length space-time mean \( \mu(a^k(x^*)) \) and variance \( \sigma^2(a^k(x^*)) \) vec-
The feature diversity regularizer is defined as:

$$ R_{\text{feat}}(x^*) = \sum_{k \in K} \| \mu(a^k(x^*)) - E(\mu(a^k(x))) \|_2 + \sum_{k \in K} \| \sigma^2(a^k(x^*)) - E(\sigma^2(a^k(x))) \|_2 $$

where $E(\cdot)$ corresponds to the expected value of representation $a(\cdot)$ for video input $x$ part of video dataset $\mathcal{X}$. Instead of requiring access to the dataset to train over $x \in \mathcal{X}$ videos, we use the Batch Normalization [22] running mean and variance to approximate the expected mean and variance as in [54]. The mean and variance estimates are then reformulated as $E(\mu(a^k(x))) \approx BN^k(\text{running mean})$ and $E(\sigma^2(a^k(x))) \approx BN^k(\text{running variance})$. An illustration of the improved search space achieved with the inclusion of feature diversity is shown in Figure 3.

3.4. Aggregation for Model Inversion

We have introduced model priming as a method to obtain a strong signal from a stimulus video with which noise-initialized videos can be optimized. To implicitly enforce transition continuity across frames, we include a temporal coherence regularizer. In addition, as the stimulus does not provide prior intuition about the diversity of features, we use a feature diversity regularizer to enhance the search space. Given their complementary properties, we formulate the final LEAPS objective as the combination of model priming, temporal coherence, and feature diversity regularizers. In line with spatial feature visualization losses that synthesize distinguishable features from noise [1, 36, 47, 64], we include a cross-entropy loss $\mathcal{L}_{\text{CE}}$ from class predictions given the synthesized input $x^*$:

$$ \mathcal{L}(x^*, v, y) = \mathcal{L}_{\text{CE}}(x^*, y) + \mathcal{L}_{\text{prim}}(x^*, v) + rR(x^*) $$

where the regularizer term combines (2) and (3): $R(x^*) = R_{\text{coh}} + R_{\text{feat}}$. Respectively, $r$ is a regularizer scaling factor. The final LEAPS objective enables the synthesis of high-fidelity features without being constrained by data availability or architecture types as shown in the results section.

4. Main Results

We first detail the train scheme and implementation settings used in Section 4.1. We then compare our proposed LEAPS method to image-based feature visualization methods that we extend to video and use as baselines in Section 4.2. We also qualitatively and quantitatively investigate the synthesized videos across a variety of video models in Section 4.3. Finally, in Section 4.4 we perform ablation studies for the LEAPS objective as well as different updatable input spatiotemporal resolutions.

4.1. Experimental details

Model details. We invert 3D [20]/(2+1)D [58]/CSN [57] ResNet-50 (R50), X3D [11], TimeSformer [4], Video Swin [29], MViTv2 [27], rev-MViT [32], and UniFormerv2 [26] networks. For all our experiments, we use the official networks available from their respective repositories pretrained on Kinetics-400 [7]. Due to its limited computational overhead and requirement of BN layers by the feature diversity regularizer in (3), we use X3D-S as the verifier network $S(\cdot)$. We note that only the internal representations and predictions from the inverted and verifier models are used. Both models only run inference and remain fixed throughout the feature synthesis.

Feature synthesis optimization details. Video input $x^*$ is initialized with size of $8 \times 224^2$. For models [11, 4, 29, 27, 32, 26] which use inputs of fixed size, we first interpolate $x^*$ to match the required size$^2$. Priming stimuli are selected from the Kinetics-400 validation set randomly. For transformer models, $x^*$ and $v$ are first tokenized. We use Adam [25] with a learning rate of 0.2 and a cosine decrease policy as in [64]. We use a total of 2K gradient updates. As in [36], we set $\delta = 1$. To discover the optimal $\lambda$ and $r$ hyperparameters for each network we use Mango [46] to perform a simple grid search for 1K gradient updates. A full overview of the hyperparameters used by each model is available in Table S1 in the supplementary material.

4.2. Baseline results

We first assess the quality of the synthesized videos. We invert 3D R50, X3D-M, and Video Swin-B models and report the averaged top-1 classification accuracies and Inception Scores (IS) [45] on synthesized videos using each video from the validation set of Kinetics-400 as a stimulus. We extend two prominent image-based feature visualization methods making them applicable to spatiotemporal models:

---

2Due to space limitations in Figures 5 to 8 the last 2 frames of $x^*$ are not shown.
Table 1: **Quantitative results for mean top-1 accuracy and Inception Score (IS).** The best results per metric are in **bold** and per architecture are underlined.

<table>
<thead>
<tr>
<th>Visualization method</th>
<th>top-1 (%) model ver.</th>
<th>Inception Score (IS) model verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3D R50</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D Deep Dream [1]</td>
<td>34.5 ± 2.6</td>
<td>1.1 ± 0.1</td>
</tr>
<tr>
<td>3D AM [13]</td>
<td>41.4 ± 5.8</td>
<td>1.4 ± 0.3</td>
</tr>
<tr>
<td>LEAPS ours (L)</td>
<td>67.9 ± 53.1</td>
<td>3.9 ± 0.9</td>
</tr>
<tr>
<td>LEAPS ours (L + R)</td>
<td>74.3 ± 60.2</td>
<td>5.1 ± 0.8</td>
</tr>
<tr>
<td>LEAPS ours (full)</td>
<td>86.2 ± 68.5</td>
<td>9.0 ± 1.0</td>
</tr>
<tr>
<td><strong>X3Dv</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D Deep Dream [1]</td>
<td>18.1 ± 1.9</td>
<td>1.1 ± 0.1</td>
</tr>
<tr>
<td>3D AM [13]</td>
<td>33.2 ± 5.6</td>
<td>1.3 ± 0.3</td>
</tr>
<tr>
<td>LEAPS ours (L)</td>
<td>73.4 ± 59.8</td>
<td>5.1 ± 1.1</td>
</tr>
<tr>
<td>LEAPS ours (L + R)</td>
<td>81.1 ± 69.3</td>
<td>8.8 ± 1.5</td>
</tr>
<tr>
<td>LEAPS ours (full)</td>
<td>90.3 ± 82.5</td>
<td>11.4 ± 0.9</td>
</tr>
</tbody>
</table>

| **Video Swin-B**     |                      |                                  |
| 3D Deep Dream [1]    | 15.9 ± 1.4           | 1.4 ± 0.2                        |
| 3D AM [13]           | 25.6 ± 2.2           | 1.6 ± 0.4                        |
| LEAPS ours (L)       | 71.2 ± 58.6          | 4.4 ± 0.8                        |
| LEAPS ours (L + R)   | 76.0 ± 65.4          | 5.3 ± 1.5                        |
| LEAPS ours (full)    | 87.4 ± 74.3          | 9.8 ± 1.3                        |

**DeepDream** [1] optimizes the input by a cross-entropy loss. It uses two regularizers including the total variance and $l_2$ norm on the input to improve convergence.

**Activation Maximization (AM) [31]** optimizes a random noise image by gradient ascent to maximize the activation of a specific class. We specifically adapt [13] for visualizing concurrent spatiotemporal representations, as it is the only prior method for visualizing features over space and time.

**Quantitative evaluation.** Table 1 shows the average top-1 accuracies and IS obtained by both the inverted models and verifier when inferring synthesized videos. For DeepDream and AM that do not use priming, the statistics are averaged across 10 runs per class. In LEAPS the statistics are calculated using each video in the Kinetics validation set as stimulus. For both measures, LEAPS yields consistently higher accuracies and IS compared to DeepDream and AM. Notably, it significantly improves the verifier accuracy across all three architectures. This demonstrates the merits of model priming as both DeepDream and AM optimize inputs solely by maximizing feature or class activations, without using the information-rich internal representations provided by a stimulus. This trend is also visible in the IS, as LEAPS regularizes the synthesized features in order to better represent learned temporally coherent motions.

**Qualitative examples.** Videos for the class *salsa spin* synthesized from different methods are shown in Figure 4. Features synthesized with LEAPS are significantly more visually distinct compared to those of baseline methods. As shown, videos produced by image-based methods extended to videos fail to represent learned spatiotemporal deep features. The visual quality of the synthesized videos also correlates with the accuracies and IS in Table 1.

Figure 5 illustrates synthesized videos from different runs given the same stimulus for class *bartending*. A substantial difference in the quality and representability of the visualizations between LEAPS and the extended 3D AM can be seen across runs. Despite the use of a video stimulus to prime the network, LEAPS visualizations are not constrained by the representations of the stimulus video. Each run of our LEAPS method shows a distinct visual style as feature diversity is encouraged during optimization through the homonym regularizer term. Effectively, LEAPS can be used as a tool for investigating the different class-specific features learned by each model.

**4.3. Analysis of LEAPS synthesized video**

The generalizability of feature visualization methods across multiple architectures is largely neglected with features from only a small subset of models being visualized.
Figure 6: **Qualitative examples of synthesized spatiotemporal features with LEAPS.** Models are primed with a stimulus video, shown on the left of each row of synthesized videos.

Figure 7: **SSIM and PSNR statistics over real and synthesized videos** using Video Swin-B features. Lower values correspond to larger cross-frame differences.

To evaluate the architecture-independent nature of LEAPS and better understand the visual fidelity of the synthesized videos, we investigate the applicability of LEAPS over a range of convolutional and attention-based video models.

**Generalizability.** The quality of the produced feature visualizations may depend on the architecture used, as models vary in terms of their complexities and feature spaces that they employ. We compare our proposed LEAPS visualization method by inverting common video models shown in Figure 6 for an arbitrary number of Kinetics classes. A common theme that arises for all models is the association of objects with specific actions. For example, the feature visualizations for the *throwing ball*, *catching/throwing frisbee*, *eating spaghetti*, and *snatch weight lifting* actions from inverted features of 3D R50, (2+1)D R50, Video Swin-B, and rev-MViT-B respectively, all optimize the video to primarily focus on the objects associated with the specific actions. Instead, for actions that are primarily perceived by motions performed, e.g. *running on treadmill* and *dancing ballet*, the actors/performers of the target actions are shown to be more conceptually influential to the model’s learned preconscious of an action, with their associated motions and movements captured by the produced visualizations. In addition, the visualizations for cases *eating spaghetti*, *blowing out candles*, and *sword fighting* reveal that networks also learn to temporally bound distinct spatiotemporal features of certain actions. This is an important ability to be learned by video models as it effectively demonstrates their capacity to filter temporal information alongside the spatial signal. Notably, LEAPS visualizations for both convolutional and attention-based models show that learned features have
tics from the stimulus video of class doing aerobics. As shown, the PSNR and SSIM statistics show the ability to represent such variations in the produced features visualizations. In contrast, the same video can be fundamentally different with both large and small changes occurring between frames. We observe that the speeds with which motions are performed within a video. We report the Peak Signal Noise Ratio (PSNR) and Small changes occurring between frames. In contrast, the same video can be fundamentally different with both large and small changes occurring between frames. In contrast, to image-based methods extended to video, LEAPS shows the ability to represent such variations in the produced feature visualizations. As shown, the PSNR and SSIM statistics from the stimulus video of class doing aerobics in Figure 7a, follow similar trends as those produced by the synthesized LEAPS video in Figure 7b.

**Video synthesis optimization.** We visualize the resulting synthesized video $x'$ at different iteration steps during optimization in Figure 8. General features such as the outlines of objects and actors as well as their movements are synthesized first by our proposed method. Interestingly, later iterations show to refine the visualized features by further synthesizing visual details. This demonstrates a level of learned hierarchy by video models as to the types of spatiotemporal learned features that are associated with a specific action. Our proposed use of model priming in tandem with temporal coherence and feature diversity regularization terms shows to enable the visualization of models’ spatiotemporal representations of actions, at a finer quality and detail.

### 4.4. Ablation studies

In this section, we conduct ablation studies reporting model statistics over synthesized videos. We initially consider the effect of the distance function used during model priming. Additionally, we report statistics over different combinations of priming and introduced regularizers across the tested architectures. Finally, we present quantitative results when using inputs of different spatiotemporal sizes alongside the resulting averaged latency times.

**Temporal coherence.** As we show in Figure 7, LEAPS can visualize motions performed over different speeds (slow/fast). We report the Peak Signal Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) for consecutive frame pairs to analyze the variations observed by both fast and slow motions within a video. We observe that the speeds with which motions are performed within the same video can be fundamentally different with both large and small changes occurring between frames. In contrast to image-based methods extended to video, LEAPS shows the ability to represent such variations in the produced feature visualizations. As shown, the PSNR and SSIM statistics from the stimulus video of class doing aerobics in Figure 7a, follow similar trends as those produced by the synthesized LEAPS video in Figure 7b.

**Video synthesis optimization.** We visualize the resulting synthesized video $x'$ at different iteration steps during optimization in Figure 8. General features such as the outlines of objects and actors as well as their movements are synthesized first by our proposed method. Interestingly, later iterations show to refine the visualized features by further synthesizing visual details. This demonstrates a level of learned hierarchy by video models as to the types of spatiotemporal learned features that are associated with a specific action. Our proposed use of model priming in tandem with temporal coherence and feature diversity regularization terms shows to enable the visualization of models’ spatiotemporal representations of actions, at a finer quality and detail.

### 4.4. Ablation studies

In this section, we conduct ablation studies reporting model statistics over synthesized videos. We initially consider the effect of the distance function used during model priming. Additionally, we report statistics over different combinations of priming and introduced regularizers across the tested architectures. Finally, we present quantitative results when using inputs of different spatiotemporal sizes alongside the resulting averaged latency times.

**Distance function**

In Table 2, we evaluate the impact of the distance functions used during model priming at (1). We test three different magnitude-based methods including $l_2$, $l_1$, and $JVS$, as well as the cosine similarity $cos$. Across all four spatiotemporal models, 3D R50, X3DM, Video Swin-B, and MViTv2-B, the magnitude-based methods perform favorably over the cosine similarity. This is due to $cos$ not taking into account the magnitude of the feature activation vectors from the synthesized and stimulus videos, which in turn limits the ability to synthesize representations relevant to the stimulus. Across magnitude-based methods, an average improvement of $+2\%$ and $+4.1\%$ to the inverted model’s (m) accuracy is observed with JVS compared to $l_2$ and $l_1$ respectively. The same trend also holds true for the verifier (v), with $+1.9\%$ and $+3.7\%$ accuracy improvements from $l_2$ and $l_1$ when using JVS. Comparatively to $l_2$ and $l_1$, JVS uses a combination of vector magnitudes and angles [15], providing a more balanced approach than magnitude-only or angle-only metrics. Therefore, we adopt $JVS$ as the priming distance method used between stimulus and synthesized feature vectors.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Distance function</th>
<th>$l_2$</th>
<th>$l_1$</th>
<th>$cos$</th>
<th>JVS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) 3D R50.</td>
<td>top-1 (m)</td>
<td>84.3</td>
<td>82.4</td>
<td>72.1</td>
<td>86.7</td>
</tr>
<tr>
<td></td>
<td>top-1 (v)</td>
<td>65.8</td>
<td>63.7</td>
<td>46.1</td>
<td>68.5</td>
</tr>
<tr>
<td></td>
<td>top-1 (v)</td>
<td>88.1</td>
<td>86.8</td>
<td>78.9</td>
<td>90.3</td>
</tr>
<tr>
<td></td>
<td>top-1 (v)</td>
<td>79.7</td>
<td>78.4</td>
<td>70.2</td>
<td>82.5</td>
</tr>
<tr>
<td>(c) Video Swin-B.</td>
<td>top-1 (m)</td>
<td>85.5</td>
<td>83.0</td>
<td>69.2</td>
<td>87.4</td>
</tr>
<tr>
<td></td>
<td>top-1 (v)</td>
<td>72.8</td>
<td>71.6</td>
<td>41.4</td>
<td>74.3</td>
</tr>
<tr>
<td></td>
<td>top-1 (v)</td>
<td>83.5</td>
<td>81.9</td>
<td>70.7</td>
<td>85.9</td>
</tr>
<tr>
<td></td>
<td>top-1 (v)</td>
<td>72.4</td>
<td>69.7</td>
<td>45.6</td>
<td>73.1</td>
</tr>
</tbody>
</table>

Table 2: **Top-1 accuracies of the inverted model (m) and verifier (v) on synthesized videos over different priming distance functions. Best results are in **bold.**
enhanced search space of $\mathcal{L} + \mathcal{R}$ as more diverse class features not in the stimulus are also explored. The combination of the priming objective with both regularizer terms for our proposed LEAPS consistently achieves the best results by a large margin compared to all other settings. LEAPS improves accuracy, for both the inverted model and verifier, as well as the quality of the generated videos based on the IS. These results further demonstrate that our proposed LEAPS optimization can be used across a range of architectures as a general spatiotemporal feature visualization method. Further qualitative examples for each network over different Kinetics classes are shown in Figures S1 to S4 in the supplementary alongside embeddings projections of LEAPS with different regularizer regimes in Section S3. **Synthesized video resolution.** Finally, we compare the accuracies, IS, and latency times when optimizing video inputs of different spatiotemporal sizes. Due to the majority of architectures requiring fixed-size inputs, we use 3D R50 and (2+1)D R50 as they are input size independent. From the top-1 (m/v) accuracies and IS summarized in Table 4, the best-performing setting across metrics is obtained with $16 \times 224^2$-sized inputs. We observe comparable performance on the temporally-reduced $8 \times 224^2$ setting with a significantly more balanced performance-to-latency. Notable decreases in accuracies and IS are observed with the potentially limited spatial resolution $8 \times 182^2$ setting, making it less suitable. Due to the significant improvements in the per-iteration latency of the $8 \times 224^2$ inputs, we adopt this setting throughout our experiments.

**Table 3: Top-1 accuracies and Inception Scores (IS) over different objectives** across different architectures and model variants. X3D5 is used as the verifier (v) for all experiments. The best results per variant are in bold. The verifier accuracy is denoted in gray for the case of using X3D5 for both model inversion and as the verifier.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Video model architectures and variants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3D (2+1D) [38]</td>
</tr>
<tr>
<td></td>
<td>IS</td>
</tr>
<tr>
<td>top-1 (m)</td>
<td>67.9</td>
</tr>
<tr>
<td>top-1 (v)</td>
<td>53.1</td>
</tr>
<tr>
<td>IS</td>
<td>0.591 / 0.913</td>
</tr>
<tr>
<td>LEAPS $\mathcal{L} + \mathcal{R}$</td>
<td>top-1 (m)</td>
</tr>
<tr>
<td>top-1 (v)</td>
<td>55.8</td>
</tr>
<tr>
<td>IS</td>
<td>0.832 / 1.205</td>
</tr>
<tr>
<td>LEAPS $\mathcal{L} + \mathcal{R}$</td>
<td>top-1 (m)</td>
</tr>
<tr>
<td>top-1 (v)</td>
<td>60.2</td>
</tr>
<tr>
<td>IS</td>
<td>1.5 ± 1.6</td>
</tr>
<tr>
<td>LEAPS $\mathcal{L} + \mathcal{R}$</td>
<td>top-1 (m)</td>
</tr>
<tr>
<td>top-1 (v)</td>
<td>68.5</td>
</tr>
<tr>
<td>IS</td>
<td>9.0 ± 1.0</td>
</tr>
</tbody>
</table>

**Table 4: Synthesized video size comparisons** based on the top-1 accuracies of the inverted model, and verifier, IS, and latency times for inference (\(J^1\)) and backprop (\(J^B\)). Best settings per architecture are in bold.

<table>
<thead>
<tr>
<th>Model</th>
<th>temp. x spatial (^2)</th>
<th>top-1 (m/v)</th>
<th>IS</th>
<th>Latency (secs) ((J^1 + J^B))</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D R50</td>
<td>8 × 182(^2)</td>
<td>83.4</td>
<td>67.2</td>
<td>81.1 ± 1.3</td>
</tr>
<tr>
<td></td>
<td>8 × 224(^2)</td>
<td>86.7</td>
<td>68.5</td>
<td>9.2 ± 1.0</td>
</tr>
<tr>
<td></td>
<td>16 × 224(^2)</td>
<td>87.0</td>
<td>68.7</td>
<td>9.8 ± 1.6</td>
</tr>
<tr>
<td>(2+1)D R50</td>
<td>8 × 182(^2)</td>
<td>75.4</td>
<td>62.9</td>
<td>4.2 ± 1.8</td>
</tr>
<tr>
<td></td>
<td>8 × 224(^2)</td>
<td>78.0</td>
<td>65.2</td>
<td>6.4 ± 1.3</td>
</tr>
<tr>
<td></td>
<td>16 × 224(^2)</td>
<td>78.5</td>
<td>65.4</td>
<td>6.6 ± 1.5</td>
</tr>
</tbody>
</table>

**5. Conclusions**

We have introduced LEAPS, a novel spatiotemporal model inversion method for visualizing the learned internal representations of networks through video synthesis. LEAPS uses a stimulus video to prime a model and iteratively optimize an input by minimizing the classification and priming loss. The resulting synthesized video visualizes learned concepts that are associated with classes without prior knowledge of the training data. During optimization, LEAPS uses two regularizers. The first enforces temporal coherence between feature transitions across frames in the updatable input. The second improves the diversity of the synthesized features through a domain-specific verifier network to enhance the search space. The proposed architecture-independent method has shown qualitatively and quantitatively that it can produce high-quality and visually coherent synthesized videos over a wide range of spatiotemporal convolutional and attention-based video models. The high classification scores and synthesized video quality metrics make LEAPS a generalizable and effective spatiotemporal feature visualization method. We believe that this first step towards learned spatiotemporal representations synthesis is a promising direction in understanding video models.

**Acknowledgments.** We use publicly available datasets and models. Research is funded by imec.icon Surv-AI-l lance project and FWO (Grant G0A4720N).
References


[27] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for...
classification and detection. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 4804–4814. IEEE, 2022. 1, 4, 6, 8


