LayerLock: Non-collapsing Representation Learning with Progressive Freezing

Goker Erdogan[†] Nikhil Parthasarathy[†] Catalin Ionescu[†] Drew A. Hudson[†]
Alexander Lerchner[†] Andrew Zisserman[†◊] Mehdi S. M. Sajjadi[†]
João Carreira[†]

[†]Google DeepMind [◊]University of Oxford

Abstract

We introduce LayerLock, a simple yet effective approach for self-supervised visual representation learning, that gradually transitions throughout training from predicting shallow features to deeper ones through progressive layer freezing. First, we make the observation that during training of video masked-autoencoding (MAE) models, ViT layers converge in the order of their depth: shallower layers converge early, deeper layers converge late. We then show that this observation can be exploited to accelerate standard MAE by progressively freezing the model according to an explicit schedule, throughout training. Furthermore, this same schedule can be used in a simple and scalable approach to latent prediction that does not suffer from "representation collapse". We apply our proposed approach, LayerLock, to both pixel and latent prediction approaches with large models of up to 4B parameters and show improvements on both semantic (action classification) and low level (depth estimation) vision tasks.

1. Introduction

The visual world is highly complex and contains regularities across many scales, ranging from low-level spatiotemporal cues about 3D shape and geometry, to highly abstract semantic information like object categories or actions. Biological systems can learn to extract these wide range of regularities in the visual stream in a mostly unsupervised way and hence generalize well to a wide variety of low to high-level visual tasks. Recent advances in self-supervised learning from video [5, 10, 16, 21, 32, 45, 50] bring us closer to building such systems, but still suffer from various issues of data efficiency and training stability among others. In this paper, we present a new self-supervised learning technique that progressively predicts higher-level representations of the input, and leads to a more stable and effective learning that improves performance on a range of low to high-level visual tasks, ranging from depth prediction to action classification.

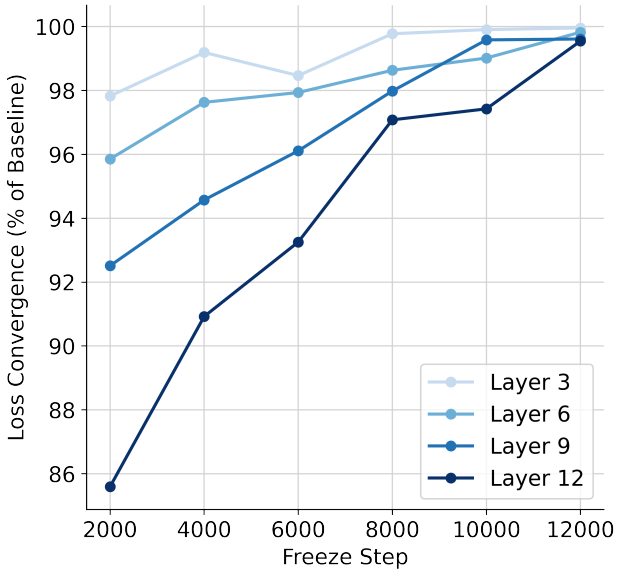

Figure 1. **In video masked auto-encoding, network layers converge during training in order of their depth.** We measure the final loss (L_{base}) of a baseline (unfrozen) model after training for 14000 steps. Each point, similarly shows the corresponding final training loss when freezing the network up to layer L at step T. We see that shallow layers "converge" faster than deeper layers as they can be frozen at earlier steps while still being able to minimize the final loss very close to L_{base}). In other words, layer convergence order is correlated with layer depth, motivating our proposed **LayerLock** progressive freezing approach to learning.

One way to categorize self-supervised visual representation learning techniques is to look at whether they make predictions in pixel space (i.e., reconstruct the input) or in a learned latent space [5, 21, 26]. Pixel prediction approaches are appealing because reconstruction provides a

stable learning signal and grounds the learned representations firmly in the visual stream. However, these approaches tend to require large amounts of training data [50], and may focus on lower-level visual information that is not always aligned with the downstream tasks we care about [2, 37]. Meanwhile, latent prediction approaches eschew predicting pixels and can learn to capture higher-level visual information and achieve good performance on downstream tasks with much less data [5, 11, 18, 32]. However, these approaches often involve various modeling tricks such as asymmetric architectures or target encoders to train without instabilities [28, 52].

In this paper, we propose LayerLock, a new approach that progressively freezes layers and dynamically updates the prediction target to transition from shallow features (pixels or early layer activations) to increasingly deeper intermediate latent model activations. LayerLock allows us to combine the best of pixel and latent-based approaches, by evolving the prediction target from low-level to high-level visual features throughout training. This progressive freezing approach is partly motivated by the observation that earlier layers in neural networks tend to converge earlier during training (see Fig. 1). This is akin to critical periods in biological development, where neural structures for specific visual abilities are only plastic for a limited time period during development [22, 27]. By dynamically evolving the prediction target throughout training, we aim to learn better video representations that capture both low and high-level visual information. This further results in a model that is highly stable, does not suffer from representation collapse issues, and allows us to train very large video models (e.g., 4B).

Contributions. Our contributions are four-fold: 1) We present LayerLock, a new simple yet effective technique that progressively freezes layers in a network and predicts a dynamically evolving target from shallow features to deeper and deeper features. 2) We show that this recipe can be applied on both pixel (e.g., VideoMAE, [45, 50]) and latent (e.g., V-JEPA [5]) prediction approaches for visual representation learning. In both cases, LayerLock learns video representations that achieve better performance on semantic tasks and low-level dense tasks. 3) We demonstrate that LayerLock saves memory and compute compared to the vanilla MAE setup, by progressively freezing intermediate layers of the network and hence requiring a backward pass over fewer and fewer layers, as training progresses. 4) We further introduce novel 3D rotary positional embeddings that lead to significant performance improvements across all the explored downstream tasks.

Paper structure. We discuss related work in the next section. In Sec. 3, we present our approach LayerLock and discuss training details, baselines, and evaluation tasks. Then, we present our experimental results in Section 4 and finally conclude in Section 5.

2. Related work

There is a wide literature on unsupervised learning from images and videos, with a wide array of approaches, which we roughly categorize into pixel vs. latent prediction approaches.

Pixel prediction. Reconstructing the input image or video, i.e. auto-encoding, has a long history [40] and been a popular representation learning technique since the early days of deep learning [24, 30, 39, 49]. While early models were mostly applied to images, there have been various video auto-encoders using convolutional architectures [25, 34, 55]. More recently, with the wide adoption of Vision Transformers [15], masked auto-encoding has emerged as the strongest pixel prediction approach for image and video representation learning [10, 21, 45, 50]. However, while pixel prediction generally leads to strong performance, it generally learns slower than latent prediction approaches [50]. There is also work that suggests predicting pixels as an objective is, in fact, not well aligned with all downstream tasks [2, 37].

Latent prediction. Latent approaches do not predict pixels and instead operate in a learned latent space [2, 26]. These approaches are appealing because pixel prediction is in general more difficult and as mentioned above may not align well with downstream tasks. An important challenge in latent prediction approaches is to avoid *representation collapse*, where the model learns to map all inputs to the same vector or other trivial prediction solution. An influential line of early work pursued contrastive predictive coding [46]; afterwards a wide variety of approaches have been proposed: variants of contrastive learning [11, 20, 44], various regularization techniques to avoid collapse in non-contrastive settings [3, 4, 54], clustering-based approaches [1, 7, 8], self-distillation [9, 32, 56], and asymmetric non-contrastive approaches [18, 19]. While initially proposed for images, most of these techniques were later applied to videos as well [13, 31, 33, 35, 36, 41, 42, 48, 51].

We propose here a simple, general, and effective method that gradually progresses from pixel to latent prediction, to achieve the best of both worlds – grounding on pixels to avoid collapse on the one hand, while progressing towards latents to encourage the emergence of abstract and semantic features on the other.

Note that one component of our approach is the progressive freezing of layers as we switch to deeper prediction targets in the network. A similar progressive freezing approach was investigated by [6] in an extended abstract for accelerating training while minimizing loss of accuracy in a supervised setting. Note that our approach LayerLock combines progressive freezing with dynamically changing the prediction targets throughout training and instead focuses more on representation learning in a self-supervised setting

rather than accelerating training.

3. Methodology

We will explain LayerLock as is applied on a standard masked-autoencoding (MAE) setup, hence we first outline the vanilla MAE approach, where we feed a masked video clip to the model and aim to reconstruct the input, essentially filling out the masked parts of the video using information from unmasked parts. We then introduce our proposed progressive freezing approach and show how we apply it on the standard MAE setup (see Fig. 2 for an illustration of our proposed model and Algorithm 1 for pseudocode outlining the forward pass). Finally, in Sec. 3.5, we will explain how we apply LayerLock to latent prediction approaches like V-JEPA.

3.1. Model Architecture

Encoding. Let $x \in \mathbb{R}^{T\times H\times W\times 3}$ be an input video clip of T frames with resolution $H \times W$. We first split this video into patches of size $t \times h \times w$ and flatten it. Then, we drop some of the input patches randomly and linearly project to D dimensions to get our context input $x_c \in \mathbb{R}^{K\times D}$. Note we do not use any form of special masking here such as tube masking and just randomly mask each patch independently of others. After adding positional embeddings to this input, we feed it to a standard Vision Transformer [14, 15] to get patch embeddings v_c.

Decoding. To predict the unmasked input x, we first append a grid of decoding latents $z \in \mathbb{R}^{N\times D}$, with one latent for each patch in the input, to the patch embeddings v_c. For flexibility and efficiency, instead of using a separate decoder network, we append z before one of the Transformer blocks in the Vision Transformer backbone. By applying the rest of the Transformer blocks over both v_c and z, we get the processed latent tokens $d \in \mathbb{R}^{N\times D}$, which are then fed to a patch-wise linear decoder. The patch-wise linear decoder projects each patch from D dimensions to the pixel space, and reshapes the output $\hat{x}$ to the input shape $T \times H \times W \times 3$. The model is trained on a simple L2 loss between unmasked inputs x and predictions $\hat{x}$.

Positional Embeddings. For positional embeddings, we use a novel, simple and effective, 3D version of rotary positional embeddings [43]. We partition the feature dimensions into 4 parts, and apply a separate 1D-RoPE embedding to 3 of these using the positions along the time, height, and width axes of the input video grid respectively, leaving the last part of features without position information. We further find that applying RoPE is less effective in the attention (as initially proposed) than after the first normalization layer at the beginning of the VIT block. We found this novel rotary positional embedding improves performance on downstream tasks (see Sec. 4 for more details).

For latent decoding tokens, we use a grid of rotary positional embeddings. Our initial experiments have shown that using learned latent tokens did not improve results, and so we directly use positional embeddings for decoding.

This completes the description of our implementation of a 'standard' MAE where the targets are pixels. We next motivate and describe LayerLock.

3.2. Ordered layerwise convergence in MAEs

In order to motivate our proposed methodological changes to the standard MAE paradigm, we first present a key observation regarding the convergence of layers during training. We start by measuring the converged loss L_{base} for the baseline model trained for 14000 steps. We then test the convergence of different layers of a 16 layer MAE based on the following simple principle: a network block up to layer L is considered "converged" at a given step T if we can freeze the network up to this layer (at step T) and continue training such that the loss is minimized to L_{base}. The discrepancy (which we can measure as a percent deviation) between the frozen network loss and L_{base} is then a measure of layer convergence at a certain point in training.

Shown in Fig. 1, given this metric, we evaluate different layers $L \in [3, 6, 9, 12]$ and freezing times $T \in [2K, 4K, 6K, 8K, 10K, 12K]$ and find that layer convergence is in fact correlated with network depth: low-level layers converge earlier in training than deeper layers. Based on these observations, we propose that MAE training can be accelerated via a schedule-based progressive freezing approach. The results of this baseline are shown in Sec. 4.

3.3. LayerLock: Progressive freezing with latent prediction

While progressive freezing can acelerate standard MAE training, freezing layers can also act as a way to generate stable latent targets, similar to how EMA networks and stop-gradients are used in methods such as V-JEPA [5]. Therefore, our final method, LayerLock extends simple layer freezing to dynamically transition from predicting shallow features (e.g., pixels or early layer activations) towards deeper latent prediction as layers are frozen. Specifically, let us assume we freeze the first k layers of the model, and predict h_k – the output of the kth layer for the full input x. To predict h_k, we take the processed latent tokens d and use a new patch-wise linear decoder to project each token to the number of features in the target layer to get predictions $\hat{h_k}$. The model is still trained on a simple L2 loss between targets h_k and predictions $\hat{h_k}$.

For freezing layers, we find the following simple schedule works well. First, we train the model on pixels as targets for N_{pixel} steps. Then every N steps, we freeze the next k unfrozen layers of the model and predict h_k, the output of the kth layer for the full input x. After N further steps of

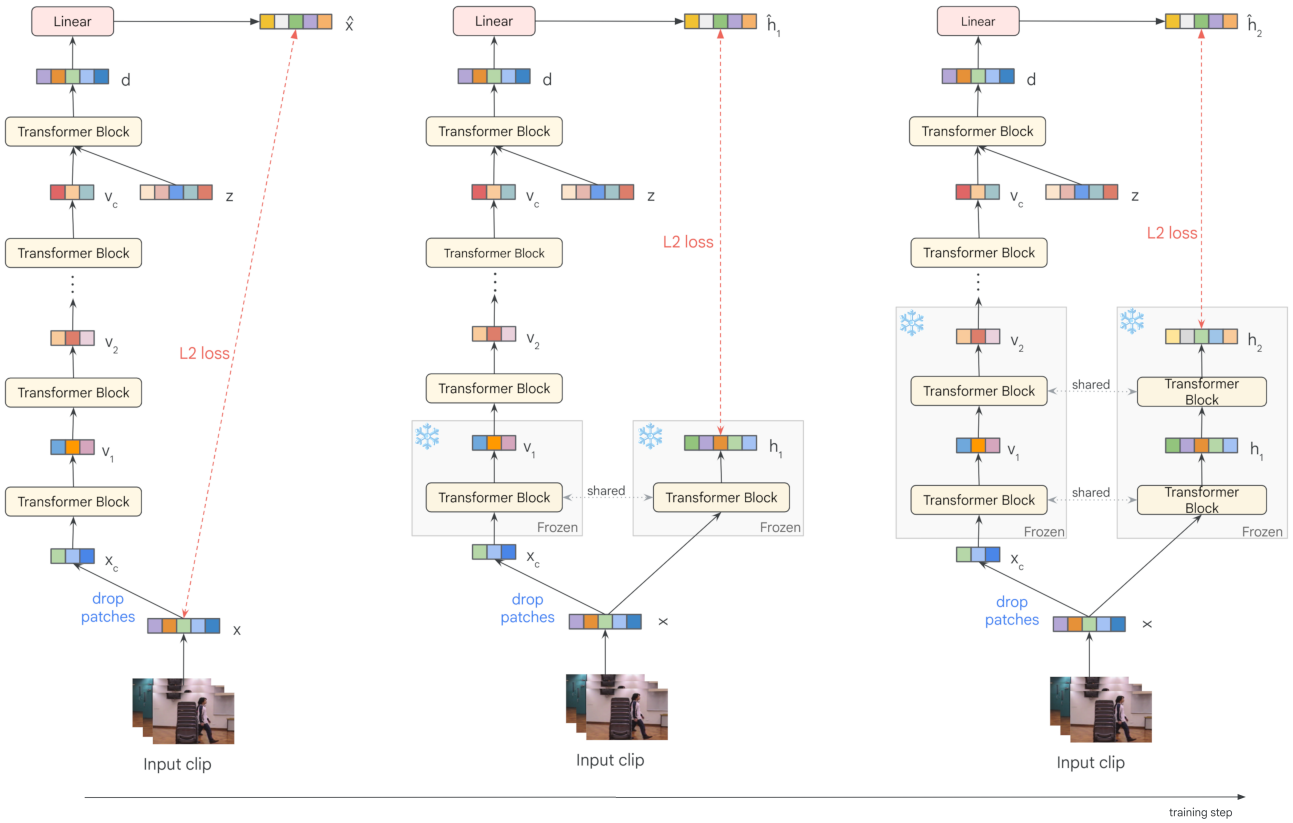

Figure 2. **Proposed learning paradigm.** Learning goes through multiple stages as the model switches between prediction targets: **Left:** no frozen layers, predicting pixels x, **Middle:** freezing first layer and predicting output of first layer h_1, **Right:** freezing first two layers and predicting the output of the second layer h_2. z are latents added for decoding. For illustration purposes, only a single Transformer block is shown after v_c and z are concatenated. Freezing continues progressively, according to a pre-determined schedule.

training, we freeze the next k layers to predict h_{2k}, h_{3k}. This is repeated, each time using a new learned linear layer to project from tokens to latent space. See Fig. 2 for an illustration of this process. Note that for efficiency, one can choose to compute the latent loss only on a subset of the input patches. In our experiments, we show results with both options, i.e., with and without patch dropping for latent loss.

3.4. Training

First, we demonstrate the effectiveness of LayerLock on the 4DS model family [10], a recent large-scale strong representative model for the video MAE approach [45, 50]. In this case, LayerLock starts by predicting pixels at the beginning of training and gradually transitions to predicting latent features at deeper layers as we progressively freeze the network.

Dataset. We use a dataset of 174M web videos, each 900-frames long on average. We sample 16 frame clips with a stride of 2 and randomly crop to 224×224 after resizing the smallest side to $1.15\times$ the resolution. We additionally apply random cropping and horizontal flipping. We use a patch size of $2 \times 16 \times 16$ and mask 95% of the patches. As mentioned above in Sec. 3, we experiment with dropping some of the input patches when computing targets for latent loss. In one set of experiments, we compute *latent loss* on a random 5% subset of all patches, while in another set of experiments, we do not drop any patches and use all of them when computing the latent loss. Note that, in contrast, for *pixel loss*, we always use all of the patches.

Training details. We train 4DS-LayerLock on 1 billion video clips on 256 TPUs-v6 using the AdamW optimizer [29]. We use a cosine learning rate schedule with a warmup and decay. We find it beneficial to have mini-warmups whenever we switch from one target to the next (see Sec. 4 for more details.) For efficiency, we train the 4DS models in *bfloat16* precision except for the layer normalization, softmax and loss computations. All weights are stored in *float32* and we use data and model parallelism. Additional details are provided in the supplementary material.

We follow the 4DS decoder architecture, using the last 4

Algorithm 1 Pseudocode of the forward pass of LayerLock as applied to MAE. Here `encoder` refers to the part of the ViT backbone until decoding latents are concatenated for decoding. Similarly, `decoder` refers to the rest of the backbone. See main text for more details.

```
def forward(video: [B, T, H, W, C], step: int):
  # Get layer to freeze according to schedule. freeze_layer = 0 means no freezing.
  freeze_layer = freeze_layer_schedule(step)

  tokens = patchify_and_flatten(video)  # [B, N, D]

  # Encode.
  tokens_enc = mask_random(tokens)  # [B, K, D]
  patch_embeddings, _ = encoder(tokens_enc, freeze_layer=freeze_layer)

  # Decode.
  decoding_tokens = rope(tokens.shape)  # [B, N, D]
  all_tokens = jnp.concatenate([decoding_tokens, patch_embeddings]
  out_tokens = decoder(all_tokens)[:, 0:N]
  pred_tokens = linear[freeze_layer](out_tokens)  # [B, N, D]

  # Compute targets.
  _, layer_outs = encode(tokens)  # Returns list of layer outputs
  layer_outs = [tokens, *layer_outs]  # Prepend pixels as first output
  layer_outs = jax.lax.stop_gradient(layer_outs)

  # Calculate loss
  loss = jnp.mean(jnp.square(pred_tokens - layer_outs[freeze_layer])
  return loss
```

layers of the Vision Transformer backbone. For progressive freezing, we determine the schedule following this simple heuristic: start the freezing when the norms of the first few layers seems to plateau (or start getting smaller when using weight decay). Then freeze layers one by one until we freeze 3/4 of all the layers at the end of training. For example, for the 4DS ViT-G model, this results in a schedule where we train with pixel loss for 160K steps and freeze 1 layer every 10K steps until we freeze 32 layers at the end of training. For rotary positional embeddings, we use a max wavelength of 10,000 and use 10%, 25%, and 25% of the input vector for encoding time, height, and width respectively.

3.5. Extending LayerLock to V-JEPA

To demonstrate the generality of our approach beyond the MAE methodology, we also show that LayerLock can be applied to video models that already leverage latent prediction. Specifically, we choose the highly popular V-JEPA [5] model. This model is also trained on masked videos but predicts latent features encoded by a teacher network rather than pixels. The teacher network here uses an exponentially weigthed moving average (EMA) of the student network to provide stable targets for prediction and is crucial for avoiding representation collapse as shown by earlier work [18].

We apply LayerLock on V-JEPA by predicting the first layer activations from the teacher network at the beginning of training and again gradually transitioning to predicting deeper layer activations as we freeze more and more layers of the encoder. For model architecture and training setup, we follow the original V-JEPA implementation as much as possible (see the supplementary material for more details) and use a ViT-L backbone for the encoder while using a separate 12 layer transformer for the decoder. For training our baseline V-JEPA and LayerLock versions, we use the Kinetics700 dataset [23] and train on 560 million video clips following the original V-JEPA paper. Note our training dataset is different from the one used in the original paper, which combined multiple Kinetics datasets into a single larger dataset. All of the data preprocessing setup is the same with the 4DS models, except in addition to the augmentations described for the 4DS models, we also use color jittering for training the V-JEPA models. We train on 256 TPUs-v3 using the AdamW optimizer [29] with a cosine learning rate schedule with a warmup and decay. We determine the freezing schedule following the simple heuristic described above, which results in a schedule where we train the full unfrozen model on predicting first layer features for 100K steps and freeze 1 layer every 6K steps afterward until we freeze all 24 layers at the end of training.

3.6. Evaluation

Since we are interested in the quality of the representations learned by LayerLock, we do not use fine-tuning for evaluations, but rather freeze the model once it is trained, and train a task-specific readout on a number of examples for each task. Given a video clip for evaluation, we add positional embeddings and feed it to our model without any masking to get the patch embeddings that are used for readouts. We get the patch embeddings from the layer positioned at 95% of the total network depth (e.g., for a 48-layer network, this corresponds to layer 45.). We use attention-based readouts, which in our experiments, performed much better than linear readouts. Depending on the task, we use a number of learned latent tokens to cross-attend to the patch embeddings and finally project to the output space of the evaluation task (see the supplementary material for more details). We train the readouts using 1.28M examples and AdamW optimizer on the evaluation task.

Tasks. We evaluate on a set of both low and high-level visual tasks from the 4DS perception tasks suite [10]: (1) action classification on Kinetics700 [23] (2) action classification on SSv2 [17], both of which require higher-level, spatio-temporal semantic understanding (3) monocular depth estimation in ScanNet [12], which test lower-level 3D perception. We provide full detail about all the tasks in the supplementary material.

3.7. Baselines

As mentioned in Sec. 3.4, we apply LayerLock on strong representatives of pixel and latent prediction approaches for video representation learning. Hence we compare LayerLock to 4DS [10] model, and V-JEPA [5] model, as representatives of the pixel and latent prediction approaches respectively. All models use a ViT backbone and tokenize videos into patches of size $2 \times 16 \times 16$.

4. Results

4.1. Progressive freezing for more efficient MAE training

Leveraging the convergence observations described in Sec. 3.2 and shown in Fig. 1, we now show that we can make MAE training more efficient without losing performance simply with progressive layer-freezing. For this experiment, we use a ViT-G backbone trained on 250M examples (same as for the ablations in Sec. 4.3).

Fig. 3 (Left) highlights that we can progressively freeze (filled bars) with negligible loss in performance compared with the baseline (dotted bars) on two tasks (56.1% vs. 56.0 accuracy on SSv2, and 0.15 vs. 0.16 relative error on ScanNet). Moreover, this competitive performance is achieved with 9% fewer total FLOPs and 16% less peak memory usage as seen in Fig. 3 (Right). We note that while this experiment was done at a shorter schedule, these efficiency gains only get larger with longer schedules- at 1B training examples, FLOP efficiency gains go up to 19%.

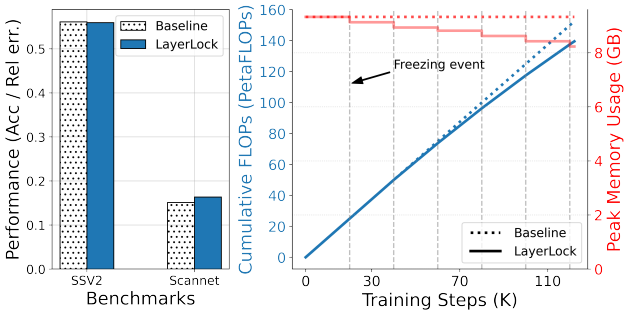

Figure 3. **Progressive freezing of layers saves on total training cost and peak memory utilization without loss in performance. (Right)** Cumulative training cost (PetaFLOPs) and peak memory usage at each step (GiB) are shown for the baseline (unfrozen) and our method. Freezing events are indicated by gray vertical lines. **(Left)** Performance of baseline unfrozen MAE (dotted) vs. progressive freezing MAE on SSV2 (accuracy) and ScanNet depth pred (rel. error).

4.2. LayerLock vs. Baselines

We begin by looking at how well our proposed model LayerLock performs compared to baseline video models in the literature. Our aim here is to show that applying our LayerLock recipe to both pixel and latent prediction approaches leads to better video representations. We evaluate each model under the same evaluation protocol described in Sec. 3 and report results in Tab. 1.

We use the following metrics for each evaluation task. For SSv2 and Kinetics700 action classification, we report Top-1 accuracy as a percentage. For ScanNet depth prediction, we use Absolute Relative Error (Lower is better and the minimum value is 0).

In Tab. 1, we show how LayerLock compares with the baseline 4DS MAE approach at two different model scales (G and e). Compared to the baseline models, those that use our LayerLock approach improve significantly on both action recognition tasks while maintaining or improving slightly on depth prediction.

To further demonstrate the generality of our method, we also show that LayerLock can also be applied to the latent prediction method, V-JEPA. We see that even in this case, LayerLock leads to significant improvements both on SSv2 and K700-2020 action recognition tasks, and ScanNet depth estimation task. In sum, these results demonstrate that LayerLock can be a general recipe for improving the video representations learned by both pixel and latent prediction approaches.

Table 1. LayerLock improves training of both pixel and latent prediction based video models when evaluated on action recognition and depth prediction tasks. **Bold** indicates best performance within the group. All results are obtained using the same evaluation protocol (see text for details). ScanNet values are multiplied by 10 for better readability.

Model	LayerLock	Size (M)	SSv2 (↑)	K700-2020 (↑)	ScanNet (↓)
4DS-G (MAE)	✗	1,848	63.1	52.1	1.02
4DS-G (MAE)	✓	1,868	**66.1**	**56.3**	**1.00**
4DS-e (MAE)	✗	3,811	64.6	54.4	**0.94**
4DS-e (MAE)	✓	3,818	**67.1**	**57.9**	0.98
V-JEPA-L	✗	235	52.1	42.5	1.57
V-JEPA-L	✓	303	**57.0**	**43.5**	**1.51**

Table 2. Ablation: We can increase the efficiency of LayerLock by using just 5% of patches for the latent-loss. This under-performs our best model but still beats the baseline significantly on SSv2. For this ablation only we use the same training settings as in Tab. 1

.

Model	Latent Loss % Patches	SSv2 (↑)	ScanNet (↓)
4DS-G (MAE)	-	63.1	1.02
4DS-G LayerLock	5%	64.9	1.12
4DS-G LayerLock	100%	66.1	1.00

Table 3. Ablation: Adding latent losses to the MAE paradigm without freezing leads to representation collapse.

Model	SSv2 (↑)	ScanNet (↓)
4DS-H	50.1	0.19
4DS-H + latent (const)	3.7	0.38
4DS-H + latent (cosine)	5.6	0.37

Table 4. Ablation: Adding 3D RoPE embeddings improves baseline models independent of LayerLock. In concert with LayerLock, we get further improvements.

Model	3D-RoPE	SSv2 (↑)	ScanNet (↓)
4DS-G	✗	56.1	0.15
4DS-G	✓	58.9	0.13
4DS-G LayerLock	✓	60.1	0.13

4.3. Ablations

We run an extensive series of ablation studies to understand why LayerLock performs better than a vanilla MAE model. Unless otherwise specified, for ablation experiments, we run a ViT-G model on 250M examples and evaluate on SSv2 action classification and ScanNet depth prediction.

Efficient latent prediction via patch subselection. The results in Tab. 1 show the effectiveness of LayerLock, but as described earlier, we also explore increasing the efficiency of the method by restricting the number of patches on which the latent loss is calculated. We see in Tab. 2 that even with *only 5% of patches* used for latent loss, LayerLock leads to performance improvements in action classification tasks while only showing a modest decrease in depth estimation performance. It will be interesting to explore this performance vs. compute trade-off in a more granular way in future work.

MAE with latent losses collapses. We've demonstrated that LayerLock benefits from two components: progressive freezing and switching from pixel reconstruction to latent prediction losses, but one might ask, can we just add latent losses to the traditional MAE? In Tab. 3, we test this using the 4DS-H MAE model (600M parameters) on 100M examples (due to resource constraints). We train on a weighted sum of latent losses (from intermediate layers) and the pixel loss. We experiment with both constant weight for the latent losses and a cosine schedule over latent loss weight. In all cases, we see serious signs of collapsing of the learned representations. This demonstrates the necessity of progressive freezing to avoid such collapse when introducing latent prediction losses.

3D rotary positional embeddings. As mentioned in Sec. 3, we use a novel 3D version of rotary positional embeddings (RoPE). Our experiments (shown in Tab. 4) show that our RoPE variant consistently improves performance for both the baseline and our LayerLock model, yielding, for instance, 2.5% improvement in SSv2 classification performance.

Single vs. multiple targets. At any point in training, LayerLock predicts only a single target (either pixels or an intermediate layer). An alternative approach would be to keep the previous prediction targets every time we switch to a new target and simply sum the losses from each target as

Table 5. Ablation: Single prediction targets from the latest frozen layer are sufficient in the LayerLock setup when compared with more complex multi-target prediction.

LayerLock Target Choice	SSv2 (↑)	ScanNet (↓)
Single (latest) target	55.5	0.16
Multiple targets	55.4	0.16

Table 6. Ablation: Effect of latent loss warmup.

Latent Loss Warmup Steps	SSv2 (↑)	ScanNet (↓)
No warmup	58.7	0.14
1K steps	59.8	0.15
3K steps	59.4	0.15

we add new training targets throughout training. This might be beneficial because the model is always trained on pixel loss in this setting and has less risk of diverging and forgetting how to predict pixels, as opposed to the single target approach. Notably, our experiments (reported in Tab. 5) reveal that the simpler approach we use in LayerLock, of always keeping the latest target, and training on a *single loss* (instead of pixel and all latent losses together) results in an equally well performing model.

Latent loss warmup. As mentioned in Sec. 3, every time we switch to predicting a new intermediate layer, we do a mini learning rate warmup where we gradually increase the learning rate. Our experiments (shown in Tab. 6) indicate that this leads to a small but consistent improvement in performance (e.g., 1% on SSv2 action classification).

Freezing schedule. In our experiments, we use a freezing schedule defined by the following parameters: *freezing start*: number of initial steps before freezing, *freezing interval*: number of steps between each freezing event, *layer jump*: number of layers to freeze at each freezing event, and *target layers*: the layers to compute latent losses for. In this study, we vary each of these parameters to understand how they affect downstream performance. Due to resource constraints, we run these experiments on a ViT-B model trained on 50M examples (see Tab. 7). First, as we vary the freezing start step, we see that fewer steps tend to lead to worse performance, suggesting that in these cases the model is not trained on pixel loss long enough (i.e., the early layers have not converged yet). Secondly, we look at how the number of layers we freeze at each freezing event (i.e, layer jump) affects performance. As we freeze more layers, we see significant drops in performance. This is likely due to more layers being frozen for longer as we increase the layer jump. Finally, we look at how the freezing interval affects performance. Note that as we increase the freezing interval, we need to adjust the layer jump to make sure we freeze the same proportion of layers in the model in each case. We see that increasing the freezing interval leads to worse performance, with the shortest freezing interval (2K steps) giving the best results. This suggests that gradual freezing of layers is more effective, even though in this case earlier layers are kept frozen for longer.

Table 7. Ablation: Effect of freezing schedule. Baseline model uses a schedule with start=6K, interval=4K, jump=2, targets=(1,3,5,7).

Model	SSv2 (↑)	ScanNet (↓)
4DS-G LayerLock	38.3	0.24
start=2K	36.0	0.25
start=4K	37.1	0.25
jump=3, targets=(2, 5, 8, 11)	36.0	0.25
jump=4, targets=(3, 7, 11)	33.1	0.26
interval=2K, jump=1	40.0	0.24
interval=8K, jump=4	36.2	0.25
interval=16K, jump=8	28.5	0.28

5. Conclusions

In this paper, we first presented an observation: when training masked-autoencoding models with ViT architectures, earlier layers in the network converge earlier than later ones. This motivated our approach LayerLock, that progressively freezes layers according to a freezing schedule and saves compute and memory while reaching the same final performance.

Secondly, following this observation, we have presented a new self-supervised video representation learning approach that trains on a dynamically evolving prediction target, starting from low-level shallow features and gradually transitioning to increasingly higher-level deeper features throughout training, as we progressively freeze the model's layers. This results in an effective representation learning technique that is more efficient than pixel prediction approaches like MAE and yet is more stable than latent prediction approaches, avoiding their common issue of representation collapse. As shown in our main results and extensive ablations, LayerLock leads to improvements in high-level visual downstream tasks like action classification, while largely maintaining performance on low-level tasks like depth estimation.

Looking ahead, we are excited about the potential compute and memory saved by progressive freezing, as it opens new avenues for scaling to longer videos, larger resolutions and even deeper models.

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. 2019. 2

[2] Randall Balestriero and Yann LeCun. Learning by reconstruction produces uninformative features for perception. *arXiv [cs.CV]*, 2024. 2

[3] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-Invariance-Covariance regularization for Self-Supervised learning. 2021. 2

[4] Adrien Bardes, Jean Ponce, and Yann LeCun. VICRegL: Self-Supervised learning of local visual features. 2022. 2

[5] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024. Featured Certification. 1, 2, 3, 5, 6

[6] Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Freezeout: Accelerate training by progressively freezing layers, 2017. 2

[7] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. 2018. 2

[8] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. 2020. 2

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[10] João Carreira, Dilara Gokay, Michael King, Chuhan Zhang, Ignacio Rocco, Aravindh Mahendran, Thomas Albert Keck, Joseph Heyward, Skanda Koppula, Etienne Pot, Goker Erdogan, Yana Hasson, Yi Yang, Klaus Greff, Guillaume Le Moing, Sjoerd van Steenkiste, Daniel Zoran, Drew A Hudson, Pedro Vélez, Luisa Polanía, Luke Friedman, Chris Duvarney, Ross Goroshin, Kelsey Allen, Jacob Walker, Rishabh Kabra, Eric Aboussouan, Jennifer Sun, Thomas Kipf, Carl Doersch, Viorica Pătrăucean, Dima Damen, Pauline Luc, Mehdi S M Sajjadi, and Andrew Zisserman. Scaling 4D representations. *arXiv [cs.CV]*, 2024. 1, 2, 4, 6

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2

[12] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 6, 2

[13] Ishan Dave, Rohit Gupta, Mamshad Nayeem Rizve, and Mubarak Shah. TCLR: Temporal contrastive learning for video representation. 2021. 2

[14] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *ICML*, 2023. 3

[15] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint*, 2020. 2, 3

[16] Alaaeldin El-Nouby, Michal Klein, Shuangfei Zhai, Miguel Angel Bautista, Alexander Toshev, Vaishaal Shankar, Joshua M Susskind, and Armand Joulin. Scalable pre-training of large autoregressive image models. *ICML*, 2024. 1

[17] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017. 6

[18] Jean-Bastien Grill, Florian Strub, Florent Altche, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. 2020. 2, 5

[19] Zhaohan Daniel Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, Michal Valko, Rémi Munos, Mohammad Gheshlaghi Azar, and Bilal Piot. BYOL-explore: Exploration by bootstrapped prediction. *arXiv [cs.LG]*, 2022. 2

[20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 2

[21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1, 2

[22] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591, 1959. 2

[23] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017. 5, 6, 1

[24] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 2

[25] Zihang Lai, Sifei Liu, Alexei A Efros, and Xiaolong Wang. Video autoencoder: self-supervised disentanglement of 3d structure and motion. In *ICCV*, 2021. 2

[26] Yann Lecun. A path towards autonomous machine intelligence version 0.9.2, 2022-06-27. https://openreview.net/pdf?id=BZ5a1r-kVsf, 2022. Accessed: 2022-8-2. 1, 2

[27] Christiaan N Levelt and Mark Hübener. Critical-period plasticity in the visual cortex. *Annual review of neuroscience*, 35 (1):309–330, 2012. 2

[28] Alexander C Li, Alexei A Efros, and Deepak Pathak. Understanding collapse in non-contrastive siamese representation learning. In *European Conference on Computer Vision*, pages 490–505. Springer, 2022. 2

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019. 4, 5

[30] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning – ICANN 2011*, pages 52–59, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. 2

[31] Jingcheng Ni, Nan Zhou, Jie Qin, Qian Wu, Junqi Liu, Boxun Li, and Di Huang. Motion sensitive contrastive learning for self-supervised video representation. 2022. 2

[32] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. 1, 2

[33] Nikhil Parthasarathy, SM Eslami, Joao Carreira, and Olivier Henaff. Self-supervised video pretraining yields robust and more human-aligned visual representations. *Advances in Neural Information Processing Systems*, 36:65743–65765, 2023. 2

[34] Viorica Patraucean, Ankur Handa, and Roberto Cipolla. Spatio-temporal video autoencoder with differentiable memory. *arXiv preprint arXiv:1511.06309*, 2015. 2

[35] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. 2020. 2

[36] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. In *CVPR*, 2021. 2

[37] Rahul Ramesh, Anthony Bisulco, Ronald W DiTullio, Linran Wei, Vijay Balasubramanian, Kostas Daniilidis, and Pratik Chaudhari. Many perception tasks are highly redundant functions of their input data. *arXiv [cs.CV]*, 2024. 2

[38] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 2

[39] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286, Bejing, China, 2014. PMLR. 2

[40] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986. 2

[41] Mohammadreza Salehi, Efstratios Gavves, Cees G M Snoek, and Yuki M Asano. Time does tell: Self-Supervised Time-Tuning of dense image representations. 2023. 2

[42] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-Contrastive networks: Self-Supervised learning from video. 2017. 2

[43] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv [cs.CL]*, 2021. 3

[44] Yonglong Tian, Olivier J Henaff, and Aaron van den Oord. Divide and contrast: Self-supervised learning from uncurated data. 2021. 2

[45] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *NeurIPS*, 2022. 1, 2, 4

[46] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. 2018. 2

[47] Sjoerd van Steenkiste, Daniel Zoran, Yi Yang, Yulia Rubanova, Rishabh Kabra, Carl Doersch, Dilara Gokay, Joseph Heyward, Etienne Pot, Klaus Greff, Drew A. Hudson, Thomas Albert Keck, Joao Carreira, Alexey Dosovitskiy, Mehdi S. M. Sajjadi, and Thomas Kipf. Moving off-the-grid: Scene-grounded video representations. In *NeurIPS*, 2024. 2

[48] Shashanka Venkataramanan, Mamshad Nayeem Rizve, João Carreira, Yuki M Asano, and Yannis Avrithis. Is imagenet worth 1 video? learning strong image encoders from 1 long unlabelled video. *arXiv preprint arXiv:2310.08584*, 2023. 2

[49] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(110):3371–3408, 2010. 2

[50] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. Videomae v2: Scaling video masked autoencoders with dual masking. In *CVPR*, 2023. 1, 2, 4

[51] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 2

[52] Xiao Wang, Haoqi Fan, Yuandong Tian, Daisuke Kihara, and Xinlei Chen. On the importance of asymmetry for siamese representation learning. *arXiv [cs.CV]*, 2022. 2

[53] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 2

[54] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. Barlow twins: Self-Supervised learning via redundancy reduction. 2021. 2

[55] Yiru Zhao, Bing Deng, Chen Shen, Yao Liu, Hongtao Lu, and Xian-Sheng Hua. Spatio-temporal autoencoder for video anomaly detection. In *Proceedings of the 25th ACM International Conference on Multimedia*, page 1933–1941, New York, NY, USA, 2017. Association for Computing Machinery. 2

[56] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. iBOT: Image BERT Pre-Training with online tokenizer. 2021. 2