Transformation-based Adversarial Video Prediction on Large-Scale Data

Pauline Luc, Aidan Clark, Sander Dieleman, Diego de Las Casas,
Yotam Doron, Albin Cassirer, Karen Simonyan
DeepMind, London, UK
{paulineluc,aidanclark}@google.com

Abstract

Recent breakthroughs in adversarial generative modeling have led to models capable of producing video samples of high quality, even on large and complex datasets of real-world video. In this work, we focus on the task of video prediction, where given a sequence of frames extracted from a video, the goal is to generate a plausible future sequence. We first improve the state of the art by performing a systematic empirical study of discriminator decompositions and proposing an architecture that yields faster convergence and higher performance than previous approaches. We then analyze recurrent units in the generator, and propose a novel recurrent unit which transforms its past hidden state according to predicted motion-like features, and refines it to handle dis-occlusions, scene changes and other complex behavior. We show that this recurrent unit consistently outperforms previous designs. Our final model leads to a leap in the state-of-the-art performance, obtaining a test set Fréchet Video Distance of 25.7, down from 69.2, on the large-scale Kinetics-600 dataset.

1 Introduction

The ability to anticipate future events is a crucial component of intelligence. Video prediction, the task of generating a plausible future sequence of frames given initial conditioning frames, has been increasingly studied as a proxy task towards developing this ability, as video is a modality that is cheap to acquire, available in abundant quantities and extremely rich in information. It has been shown to be useful for representation learning [62, 72, 69] and in the context of reinforcement learning, to define intrinsic rewards [7], or for planning and control [70, 19, 74].

Recently, large-scale adversarial generative modeling of images has led to highly realistic generations, even at high resolutions [33, 6, 34]. Modeling of video [63, 57] has also seen impressive advances. Clark et al. [12] showed strong results on class-conditional generation of Kinetics-600, but the setting of video prediction was found to be surprisingly difficult in comparison. This may be due to the mode dropping behavior of GANs. Indeed, generating a video from scratch allows the network to learn just a few plausible types of sequences instead of needing to infer a plausible continuation for any sequence.

In this work we focus on improving DVD-GAN-FP [12] in two ways. First, following the authors' observation that the architecture of the discriminator is key for efficient training on large-scale datasets, we propose alternative decompositions for the discriminator and conduct an empirical study, validating a new architecture that achieves faster convergence and yields improved video quality.

Second, we draw inspiration from recent state-of-the-art approaches for video prediction [24, 21] that predict the parameters of transformations, use them to warp the conditioning frames, and combine the result with direct generation. On the one hand, transformation-based prediction can use information in the conditioning frames, bypassing the need to learn to decompress representations of the relevant

elements. On the other, in the presence of dis-occlusion, of appearing objects, or of new regions of the scene unfolding due to camera motion, warping past information seems an overly complex task in comparison with generating these pixel values directly. The two approaches are hence highly complementary. We seek to incorporate such transformations in the DVD-GAN-FP architecture, to provide sufficient flexibility to model camera and object motion in a straightforward fashion, while retaining scalability of the overall architecture. For this purpose, we introduce the Transformation-based Spatial Recurrent Unit (TSRU), a novel recurrent unit that predicts motion-like features, and uses them to warp its past hidden state. It then refines the obtained predictions by generating features to account for newly visible areas of the scene, and finally fuses the two streams.

To summarize, our contributions are as follows:

- We conduct a systematic performance study of discriminator decompositions for DVD-GAN-FP and propose an architecture that achieves faster convergence and yields higher performance than previous approaches.
- We propose a novel transformation-based recurrent unit to make the generator more expressive. This module is better suited for large-scale training than existing transformation-based alternatives, and we show that it brings substantial performance improvements over classical recurrent units.
- Our final approach, combining these two improvements, leads to a large improvement on the Kinetics-600 video prediction benchmark. Qualitatively, our model yields highly realistic frames and motion patterns.

2 Background and related work

2.1 Video generation and GANs

Since the introduction of the next frame prediction task [53, 62], video prediction and generation have become increasingly popular research topics. An important line of work has focused on extending ideas from generative modeling of images to these tasks [47, 69, 79, 2, 14, 32].

Generative Adversarial Networks (GANs) [23] define a minimax game between a *discriminator* $\mathcal{D}$ learning to distinguish real data from generated samples, and a *generator* $\mathcal{G}$ learning to minimize the likelihood of the discriminator classifying generated samples as generated. In this work, we base our network architecture off DVD-GAN-FP [12], which has built on the design principles combined in BigGAN for stable, large-scale training of image generative adversarial networks [6, 49, 48, 42, 17]. The generator of DVD-GAN-FP is primarily a 2D convolutional residual network with convolutional recurrent units interspersed between blocks at multiple resolutions to model consistency between frames. The discriminator is composed of a spatial discriminator, assessing individual frames, and a spatio-temporal discriminator, assessing the concatenation of the conditioning and generated frames. We provide an overview of DVD-GAN-FP in the appendix.

Many GAN approaches decompose the discriminator into multiple subnetworks. This includes multi-resolution approaches used for images [16, 80, 27], speech [5, 38] and video [63, 12]. Some of these methods decompose the discriminator in a way which requires the existence of multiple generators, potentially sharing some hidden layers [57, 33]. While an important research direction, for the purposes of our analysis of discriminators, we do not consider such variants.

Progress on the task of video prediction has spanned multiple directions. A number of works advocate for disentangling the factors of variation in the representations used by the models through careful design of the loss and network architecture: for example, between motion and content [63, 65], pose and appearance [15] or foreground and background [69]. Object-centric representations have also been proposed [37, 82]. Some focus on alleviating error accumulation via architectural improvements [51, 66, 68]. Several works also motivate the use of various reconstruction, ranking and perceptual losses [47, 30, 43, 40, 76]. Finally, another line of work reformulates video prediction in other feature spaces than the pixel space, such as high level image features [62, 67], optical flow [71], dense trajectories [72] and segmentation maps [45, 46]. Jayaraman et al. [31] operate in the pixel space, but choose not to commit to a fixed time offset between the conditioning frames and the prediction.

2.2 Kernel-based and vector-based transformations

Tranformation-based models for video prediction rely on differentiable warping functions to generate future frames as a function of past frames. Following the terminology introduced by Reda et al. [54], they fall into one of two families of methods: *vector-based* or *kernel-based*.

Vector-based approaches consist in predicting the coordinates of a grid used to differentiably sample from the input [29]. This kind of approach has been used in conjunction with optical flow estimation [53, 52, 54]. Liu et al. [44] extend this idea, predicting a *3D pseudo-optical flow field* used to sample from the entire sequence of conditioning frames via tri-linear interpolation.

Kernel-based approaches predict the parameters of dynamic, depth-wise locally connected layers [79, 20, 68, 54]. These predicted parameters can be shared across spatial locations, forcing each spatial position to undergo the same transformation. To allow for varying motion across locations, Xue et al. [79] use several such transformations, and combine them using a predicted vector of weights at each position, to yield per-pixel filters. Predicting a fixed number of depth-wise transformation kernels alongside a per-spatial-location weighting over transformations can be seen as a factorized version of predicting a full depth-wise transformation at each position. We therefore refer to the two approaches as *factorized* and *pixelwise*. Finn et al. [20] explore both and find them to perform similarly. We refer to the appendix for a more formal description and an illustration for each.

2.3 Spatial recurrent units for video processing

Shi et al. [59] and Ballas et al. [4] introduce convolutional variants of the Long Short-Term Memory (LSTM) [26] and Gated Recurrent Unit (GRU) modules [11]. Our proposed recurrent unit builds on the convolutional recurrent units employed by DVD-GAN-FP, Convolutional GRUs (ConvGRUs), which produce output h_t based on input x_t and previous output h_{t-1} as:

$$r = \sigma(W_r \star_k [h_{t-1}; x_t] + b_r) \tag{1}$$
$$h'_t = r \odot h_{t-1} \tag{2}$$
$$c = \rho(W_c \star_k [h'_t; x_t] + b_c) \tag{3}$$
$$u = \sigma(W_u \star_k [h_{t-1}; x_t] + b_u) \tag{4}$$
$$h_t = u \odot h_{t-1} + (1 - u) \odot c \tag{5}$$

Here σ is the sigmoid function, ρ is the chosen activation function, $\star_k$ represents a $k \times k$ convolution, and the $\odot$ operator represents elementwise multiplication.

With similar motivations to ours, Xu et al. [77] propose a recurrent unit for structured video prediction, relying on dynamic prediction of the weights used in the Convolutional LSTM (ConvLSTM) update equations. To avoid over-parameterization, the authors propose a shared learned mapping of feature channel differences to corresponding $2D$ kernels. This requires processing $\mathcal{O}(C^2)$ inputs, where C is the number of channels of the hidden state, which is prohibitively expensive in our large-scale setting. We refer the interested reader to the appendix for a more detailed discussion.

Shi et al. [60] introduce TrajGRU, which also warps the past hidden state to account for motion, using vector-based transformations, and provides the result as input to all gates of a ConvGRU. Vector-based transformations allow modeling of arbitrarily large motions with a fixed number of parameters, but they require random memory access, whereas modern accelerators are better suited for computation formulated in terms of matrix multiplications. As a result, in our work, we employ kernel-based transformations. We first propose a straightforward kernel-based extension of TrajGRU, called K-TrajGRU. We then formulate a novel transformation-based recurrent unit that is simpler and more directly interpretable.

3 Large-scale transformation-based video prediction

3.1 Discriminator decompositions

Architectures that can effectively leverage a large dataset like Kinetics-600 require careful design. While the generator must create temporally consistent sequences of plausible frames, the discriminator must also be able to assess both temporal consistency and overall plausibility.

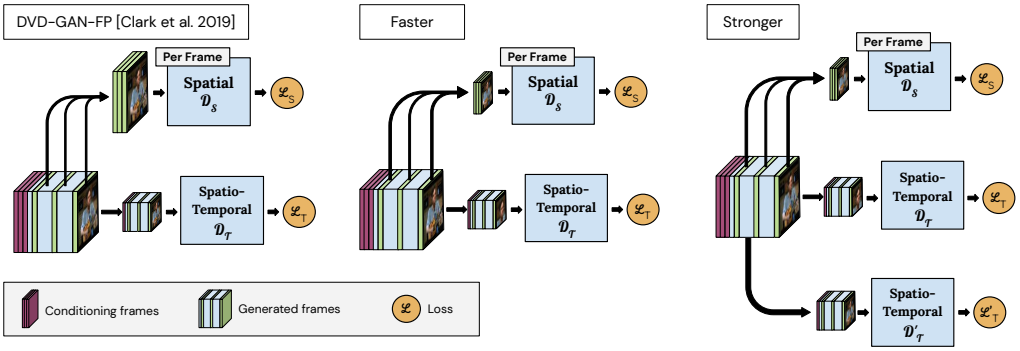

Figure 1: A visualization of the inputs each discriminator must judge. From left to right: the original DVD-GAN-FP discriminator decomposition processes full resolution frames and downsampled videos; our proposed *faster* decomposition processes downsampled frames and cropped videos; our proposed *stronger* decomposition additionally processes downsampled videos.

Both Tulyakov et al. [63] and DVD-GAN-FP [12] employ two discriminators: a *Spatial Discriminator* $\mathcal{D}_S$ and a *Temporal Discriminator* $\mathcal{D}_T$, which respectively process individual frames and video clips. While Tulyakov et al. [63] motivate the use of $\mathcal{D}_S$ mainly by improved convergence, DVD-GAN-FP makes the two discriminators complementary in order to additionally decrease computational complexity. In DVD-GAN-FP, $\mathcal{D}_S$ assesses single frame content and structure by randomly sampling K full-resolution frames and judging them individually, while $\mathcal{D}_T$ is charged with assessing global temporal structure by processing videos spatially downsampled by a factor s.

To further improve efficiency, we propose an alternative split of the roles of the discriminators, with $\mathcal{D}_S$ judging per-frame global structure, while $\mathcal{D}_T$ critiques local spatio-temporal structure. We achieve this by downsampling the K randomly sampled frames fed to $\mathcal{D}_S$ by a factor s, and cropping $T \times H/s \times W/s$ clips inside the high resolution video fed to $\mathcal{D}_T$, where T, H, W correspond to time, height, and width dimension of the input. Compared with DVD-GAN-FP, this reduces the number of pixels to process per video, from $K \times H \times W + T \times H/s \times W/s$ to $(K+T) \times H/s \times W/s$. We call this decomposition DVD-GAN-FP$_{faster}$. In practice, we follow DVD-GAN-FP and choose $s = 2$.

The potential blind spot of our new decomposition is global spatio-temporal coherence. For instance, if the motion of two spatially distant objects is correlated, then this decomposition will fail to teach $\mathcal{G}$ that one object must move based on the movement of the other. To address this limitation, we investigate a second configuration which consists in combining our proposed decomposition with a second temporal discriminator $\mathcal{D}_T^{'}$ assessing the quality of the downsampled generated videos, like in DVD-GAN-FP, and call this decomposition DVD-GAN-FP$_{stronger}$. We summarize these configurations in Figure 1 and Table 1.

3.2 Transformation-based Recurrent Units

3.2.1 Stacked kernel-based warping for efficient, multi-scale modeling of motion

As motivated in Section 2.3, we favour kernel-based transformations over vector-based ones. We rely on the stacking of multiple such transformations in the feature hierarchy of the generator to allow the modeling of large motion in a multi-scale manner, while keeping the number of parameters under control.

We first study a kernel-based extension of TrajGRU, whose equations we report in the appendix. Inspired by recent work that leverages complementary pixel-based generations and transformation-based predictions [24, 21], we also propose a novel recurrent unit whose design is simpler and more directly interpretable, which we describe next.

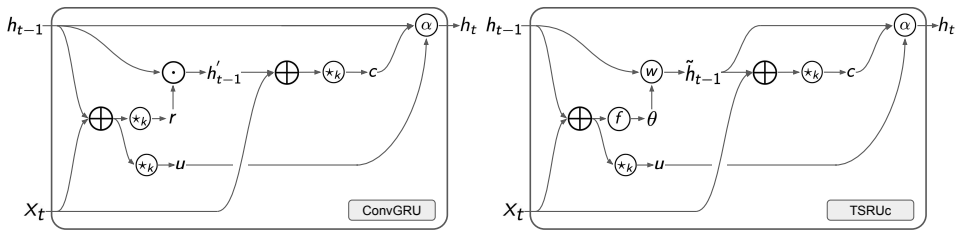

Figure 2: Information flow for ConvGRU (left) and our proposed TSRU$_c$ (right); where α represents elementwise convex combination with coefficient provided by u; $\star_k$, convolution with kernel size k; $\bigoplus$, concatenation and $\bigodot$, elementwise multiplication. Other notations follow from Eqs (6)-(10).

3.2.2 TSRU

To combine transformation-based prediction with direct generation, we begin by observing that the final equation of ConvGRU, Eq (5), already combines past information h_{t-1} with newly generated content c. Similarly, our recurrent unit combines $\tilde{h}_{t-1}$, obtained by warping h_{t-1}, with newly generated content c. Instead of computing the reset gate r and resetting h_{t-1}, we compute the parameters of a transformation θ, which we use to warp h_{t-1}. The rest of our model is unchanged (with $\tilde{h}_{t-1}$ playing the role of h'_t in c's update equation, Eq (3)). Finally, our module is described by the following equations:

$$\theta_{h,x} = f(h_{t-1}, x_t) \tag{6}$$

$$\tilde{h}_{t-1} = w(h_{t-1}; \theta_{h,x}) \tag{7}$$

$$c = \rho(W_c \star_k [\tilde{h}_{t-1}; x_t] + b_c) \tag{8}$$

$$u = \sigma(W_u \star_k [h_{t-1}; x_t] + b_u) \tag{9}$$

$$h_t = u \odot \tilde{h}_{t-1} + (1 - u) \odot c, \tag{10}$$

where f is a shallow convolutional neural network, w is a warping function described next, and the rest of the notations follow from Eqs (1)-(5). We have designed this recurrent unit to closely follow the design of ConvGRU and call it TSRU$_c$. We provide an illustration in Figure 2. We study two alternatives for the information flow between gates. TSRU$_p$ computes θ, u and c in parallel given x_t and h_{t-1}, yielding the following replacement for Eq (8): $c = \rho(W_c \star_k [h_{t-1}; x_t] + b_c)$.

At the other end, TSRU$_s$ computes each intermediate output in a fully sequential manner: like in TSRU$_c$, c is given access to $\tilde{h}_{t-1}$, but additionally, u is given access to both outputs $\tilde{h}_{t-1}$ and c, so as to make an informed decision prior to mixing. This yields the following replacement for Eq (9): $u = \sigma(W_u \star_k [\tilde{h}_{t-1}; c] + b_u)$. We also illustrate these variants in the appendix.

The resulting modules are generic and can be used as drop-in replacements for other recurrent units, which are already widely used for generative modeling of video. Provided that the conditioning frame encodings are used as initialization of its hidden representation, as in the DVD-GAN-FP architecture, this unit can predict and account for motion in the use it makes of past information, if this is beneficial. When interleaved at different levels of the feature hierarchy in the generator, like in DVD-GAN-FP, motion can naturally be modeled in a multi-scale approach. The modules are designed in such a way that a pixel-level motion representation can emerge, but we do not enforce this. In particular, we do not provide any supervision (e.g. ground truth flow fields) for the predicted motion-like features, instead allowing the network to learn representations in a data-driven way.

We study both factorized and pixelwise kernel-based warping (*c.f.* Section 2.2 and appendix). Specifically, for factorized warping, we predict a sample-dependent set $\mathbf{w}$ of N $2D$-convolution kernels of size $k \times k$ and a selection map $\mathbf{S}$, where each spatial position holds a vector of probabilities over the N warpings, which we use as coefficients to linearly combine the basis weight vectors into per-pixel weights. This map is intended to act as a pseudo-motion segmentation map, splitting the feature space into regions of homogeneous motion. In practice, we choose $N = k^2$ and $k = 3$. Pixelwise kernel-based warping consists in predicting a $k \times k$ weight vector for each spatial position, which is used locally to weigh the surrounding values of the input in a weighted average. We denote this approach by Pixelwise Transformation-based Spatial Recurrent Unit (PTSRU).

4 Experiments and Analysis

4.1 Experimental set up

Datasets Kinetics is a large dataset of YouTube videos intended for action recognition tasks. There are three versions of the dataset with increasing numbers of classes and samples: Kinetics-400 [35], Kinetics-600 [9] and Kinetics-700 [10]. Here we use the second, Kinetics-600, to enable fair comparison to prior work. This dataset contains around 500,000 10-second videos at 25 FPS spread across 600 classes. Kinetics is often used in the action recognition field, but has only recently been used for generative purposes. Li et al. [41] and Balaji et al. [3] used filtered and processed versions of the dataset, and more recently the entire unfiltered dataset has been used for generative modeling with GANs as well as autoregressive models [12, 75]. Following these works, we resize all videos to 64×64 resolution and train on randomly-selected 16 frame sequences from the training set. Specifically, $\mathcal{G}$ is trained to predict 11 frames conditioned on 5 input frames. Unfortunately, the Kinetics dataset in general is not covered by a license which would allow for showing random data samples in a paper. For this reason, for all qualitative analysis and shown samples in this work, we use conditioning frames taken from the UCF-101 dataset [61], though the underlying models are trained completely on Kinetics-600. Finally, we extend our analysis on UCF-101 [61] and BAIR Robot Pushing dataset [18], presented in the appendix.

Metrics Traditional measures for video prediction models such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) [73] rely on the existence of ground truth continuations of the conditioning frames and judge the generated frames by their similarity to the real ones. While this works for small time scales with near deterministic futures, these metrics fail when there are many potential future outcomes. Unterthiner et al. [64] propose the Fréchet Video Distance (FVD), an adaption of the Fréchet Inception Distance (FID) metric [25] which judges a generative model by comparing summary statistics of a set of samples with a set of real examples in a relevant embedding space. For FVD, the logits of an I3D network trained on Kinetics-400 [8] are used as the embedding space. We adopt this metric for ease of comparison with prior work. We also report the Inception Score (IS).

Training details All models were trained on 32 to 512 TPUv3 accelerators using the Adam [36] optimizer. In all our experiments, we validate the best model according to performance on the training set. Like BigGAN [6], we find it important to use cross-replica batch statistics where the per-batch mean and variance are calculated across all accelerators at each training step. More details on our training setup are provided in the appendix. We will make the code[1] available, as well as models trained on UCF-101 and BAIR.

4.2 Comparing Discriminator Decompositions

We begin by comparing several discriminator decompositions, including a MoCoGAN-like decomposition [63], DVD-GAN-FP [12], and our proposed decompositions, on the Kinetics video prediction benchmark [75]. We fix the number of frames K sampled for the spatial discriminator to 8. For MoCoGAN-like, we also evaluate the authors' original setting, where $K = 1$. Finally, for each of the discriminator decompositions, we evaluate a baseline where only the temporal discriminator is used, denoted respectively by *Clip* (for MoCoGAN-like), *Downsampled* (for DVD-GAN-FP), *Cropped* (for DVD-GAN-FP$_{faster}$) and *Cropped & Downsampled* (for DVD-GAN-FP$_{stronger}$).

Because we are interested in architectures which improve real experimentation time, we fix the step budget for each method according to its average step duration, measured over 30 seconds for a batch size of 512, and make sure that we do not account for any lag introduced by data loading or preprocessing. For these normalized step budgets, training takes approximately 103 hours, corresponding to $1M$ iterations for the fastest one. We summarize step budgets and results in Table 1.

For all four discriminator decompositions, the addition of a frame discriminator yields large improvements in FVD, along with improving convergence speed as previously noted by Tulyakov et al. [63]. We also observe that the *Downsampled* and *Cropped* baselines obtain poor performance compared to

[1] Our training script is heavily tied to proprietary code, and there is no simple way to release it. However, upon acceptance, we will release functions relating to data processing, models, losses and metrics.

Table 1: A comparison of different discriminator decompositions for video prediction trained for a fixed time budget evaluated on the Kinetics 600 validation set. We describe each in terms of the types of sub-discriminators it contains: frame-level (F), downsampled frame-level (dF), clipped video (clV), downsampled video (dV) and cropped video (crV). Average step duration in milliseconds (ms) and maximum memory consumption (GB) are measured for a single forward-backward training step, averaged over 30 seconds. We report the step budget for each run in number of thousands of steps (K).

	F	dF	clV	dV	crV	IS	FVD	K	ms	GB
Clip			✓			10.9	59.6	758	488	3.49
Downsampled				✓		8.9	122.1	1000	370	3.01
Cropped					✓	9.4	159.9	997	371	3.10
Cropped + Downsampled				✓	✓	11.5	44.1	792	467	3.30
MoCoGAN-like $K = 1$	✓		✓			11.3	47.4	651	568	3.95
MoCoGAN-like $K = 8$	✓		✓			11.4	46.2	561	660	3.95
DVD-GAN-FP (ours)	✓			✓		10.9	55.0	689	537	3.53
DVD-GAN-FP$_{faster}$		✓			✓	11.7	46.1	817	453	3.54
DVD-GAN-FP$_{stronger}$		✓		✓	✓	**11.8**	**39.3**	675	548	3.65

the other approaches, consistent with our observation that the *Downsampled* baseline can only assess global spatio-temporal structure, and the *Cropped* baseline can only assess local structure. In contrast, a combination of the two (*Cropped + Downsampled*) largely improves on the *Clip* baseline, which has access to all conditioning and predicted frames, and obtains improved performance with respect to the two MoCoGAN-like variants, while being 29% faster to train. Next, we find that in this setting, DVD-GAN-FP yields an improvement in terms of average step duration over the MoCoGAN-like baselines, but lags behind in terms of prediction quality. As expected, the alternative decomposition we propose has a much shorter average step duration than the MoCoGAN-like and DVD-GAN-FP baselines, and additionally closes the performance gap with MoCoGAN-like. Adding a spatial discriminator to the *Cropped & Downsampled* baseline yields our final proposed decomposition, DVD-GAN-FP$_{stronger}$, which obtains a substantial improvement over previous approaches.

4.3 Comparing Recurrent Units

We now report a comparison between recurrent units in Table 2. Here, we use the fastest of our discriminator decompositions, DVD-GAN-FP$_{faster}$, so as to maximize the number of training steps which can be afforded for each configuration in this large-scale setting. We train all models for 500k steps. Our setup is otherwise identical to the one presented in Section 4.2. We find that all our proposed transformation-based approaches perform better than using ConvGRU or ConvLSTM, suggesting that the increased expressivity of the generator is beneficial for generation quality, with the best results observed for TSRU$_p$ in this setting. Interestingly, we observe that PTSRU performs slightly worse than its TSRU$_s$ counterpart (which is the least competitive of our TSRU designs); suggesting that the factorized version may be more effective at predicting structured, globally consistent motion-like features. As a result, we do not investigate other versions of the PTSRU design. Finally, we observe a small performance improvement for all TSRU compared with K-TrajGRU, suggesting that its simplified design facilitates learning.

4.4 Scaling up

Putting it all together, we combine our strongest decomposition DVD-GAN-FP$_{stronger}$ with TSRU$_p$. We call this model TrIVD-GAN-FP, for "Transformation-based & TrIple Video Discriminator GAN". We increase the channel multiplier (described in the appendix) from 48 to 120 and train this model for 700,000 steps. We report the results of this model evaluated on the test set of Kinetics 600 in Table 3, together with the results reported by Weissenborn et al. [75] and Clark et al. [12]. We also provide unbiased estimates of standard deviation for 10 training runs. In comparison with the strong DVD-GAN-FP baseline, our contributions lead to a substantial decrease of FVD.

Table 2: Comparison of recurrent unit designs for video prediction on Kinetics 600 validation set. All models are trained for 500k steps.

	IS (↑)	FVD (↓)
ConvLSTM	11.4	51.3
ConvGRU	11.4	49.4
K-TrajGRU	11.6	45.9
TSRU_c	11.6	44.2
TSRU_p	11.5	**43.7**
TSRU_s	11.6	45.8
PTSRU_s	11.5	47.1

Table 3: Video prediction performance on Kinetics-600 test set, without frame skipping. FVD is computed using test set video statistics. We predict 11 frames conditioned on 5 input frames. We provide unbiased estimates of standard deviation over 10 runs.

Method	IS (↑)	FVD (↓)
Video Transformer	–	170 ± 5
DVD-GAN-FP	–	69.15 ± 1.16
TrIVD-GAN-FP	12.54 ± 0.06	$\mathbf{25.74 \pm 0.66}$

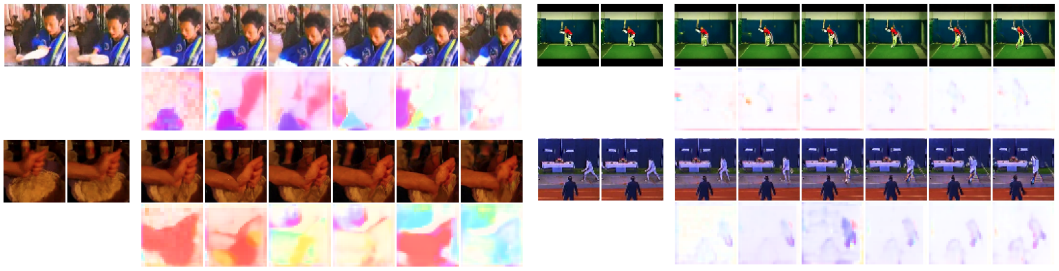

Figure 3: Qualitative evaluation on UCF-101 test set 1. Top: predictions. Second row: motion field obtained by combining the two lower levels of predicted motion-like features. All sequences are temporally subsampled by two for visualization.

In Figure 3, we show predictions and corresponding motion fields from our final model TrIVD-GAN-FP. We select them to demonstrate the interpretability of these motion fields for a number of samples. We obtain them by interpreting the local predicted weights for a given spatial position as a probability distribution over flow vectors corresponding to each of the kernel grid positions and taking the resulting expected flow value. They can be combined across resolutions by iteratively upsampling a coarse flow field, multiplying it by the upsampling factor to account for the higher resolution, and refining it with the next lower level flow field. These predictions have been obtained on UCF-101 and also show strong transfer performance. Though this is not entirely surprising given the similarity between the classes and content, it further emphasizes the generalization properties of our model. We provide an extended qualitative analysis including additional samples in the appendix.

Finally, we perform additional experiments on the BAIR Robot Pushing dataset, and on UCF-101, to demonstrate the generality of our method. We further show that our model is able to generate diverse predictions on the BAIR dataset. We report the results in the appendix.

5 Conclusion

Effectively training video prediction models on large datasets requires scalable and powerful network architectures. In this work we have systematically analyzed several approaches towards reducing the computational cost of GAN discriminators and proposed DVD-GAN-FP$_{faster}$ and DVD-GAN-FP$_{stronger}$, which improve wall-clock time and performance over the strong baseline DVD-GAN-FP. We have further motivated and proposed a family of transformation-based recurrent units – drop-in replacements for standard recurrent units – which further improve performance. Combining these contributions led to our final model, TrIVD-GAN-FP, which obtains large performance improvements and is able to make diverse predictions. For future directions, we plan to investigate the use of these recurrent units for video processing tasks, as well as the usefulness of the representations that our models learn.

Broader Impact

Video generative modeling is a well-established research problem, which can drive the development of general purpose technologies for modeling distributions of data arising from complex spatio-temporal phenomena. Kinetics-600 is a large, diverse and complex dataset, and as such, it is reasonable to expect that approaches that yield improvements on this dataset should generally prove useful in many other settings. Our proposed discriminator decomposition yields faster convergence and better performance. Our proposed transformation-based recurrent units provide additional flexibility to the generator, and are designed such that pixel-level motion representations can emerge in a data-driven way. Such advances could be applied to the acceleration of climate models. Extensions of video prediction models could replace components for which physical counterparts are too computationally intensive or too uncertain. For example, high resolution cloud simulation [22] and prediction of short-term changes in the sea ice extent using satellite images have both been identified as a "high leverage" machine learning applications for tackling climate change [55]. Our improvements could also generally benefit applications that involve decision-making in the real world, such as autonomous driving and robotics.

On a more pessimistic note, progress in the field of video generation models might be applied by malicious agents, to forge false, plausible elements in support of deceptive affirmations, thus contributing to the creation, confirmation and spread of beliefs. This can, in turn, have dramatic consequences. As the field progresses, methods that can detect generated images or videos automatically, such as [1, 56, 78], are therefore becoming increasingly important. Nguyen et al. [50] provide a survey of such methods.

Using Kinetics as a benchmark might also inadvertently result in researchers overlooking privacy concerns, motivated by research incentives. Kinetics is a dynamic dataset, with a small number of videos occasionally removed, following their removal from YouTube [35]. We believe that releasing frames or continuations for this dataset would therefore be unethical. The same concern applies to releasing generative models trained on this dataset, since we cannot guarantee that they will not be able to partially memorize some of the removed videos. To mitigate this risk, we hope that future research can continue to follow our evaluation procedure, when using this dataset, by presenting only aggregated metrics of performance, and by employing other datasets to demonstrate qualitative performance.

Acknowledgements

We thank Jean-Baptiste Alayrac, Lucas Smaira, Jeff Donahue, Jacob Walker, Jordan Hoffman, Carl Doersch, Joao Carreira, Andy Brock, Yusuf Aytar, Matteusz Malinowski, Karel Lenc, Suman Ravuri, Jason Ramapuram, Viorica Patraucean, Simon Osindero and Chloe Rosenberg for their help, feedback and insights.

References

[1] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. Mesonet: a compact facial video forgery detection network. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7. IEEE, 2018.

[2] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *International Conference on Learning Representations*, 2018.

[3] Yogesh Balaji, Martin Renqiang Min, Bing Bai, Rama Chellappa, and Hans Peter Graf. Tfgan: Improving conditioning for text-to-video synthesis. 2018.

[4] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv:1511.06432*, 2015.

[5] Mikołaj Bińkowski, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C. Cobo, and Karen Simonyan. High fidelity speech synthesis with adversarial networks. In *International Conference on Learning Representations*, 2020.

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *International Conference on Learning Representations*, 2019.

[7] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[8] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the Kinetics dataset. In *International Conference on Computer Vision*, 2017.

[9] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about Kinetics-600. *arXiv:1808.01340*, 2018.

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

[11] Kyunghyun Cho, Bart van Merrienboer, Caglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing*, 2014.

[12] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets. In *arxiv:1907.06571*, 2019.

[13] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *Neural Information Processing Systems*, 2017.

[14] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, 2018.

[15] Emily L Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In *Neural Information Processing Systems*, 2017.

[16] Emily L Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a Laplacian pyramid of adversarial networks. In *Neural Information Processing Systems*, 2015.

[17] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *International Conference on Learning Representations*, 2017.

[18] Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. In *Conference on Robot Learning*, 2017.

[19] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *ICML Deep Learning Workshop*, 2015.

[20] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Neural Information Processing Systems*, 2016.

[21] Hang Gao, Huazhe Xu, Qi-Zhi Cai, Ruth Wang, Fisher Yu, and Trevor Darrell. Disentangling propagation and generation for video prediction. In *International Conference on Computer Vision*, 2019.

[22] Pierre Gentine, Mike Pritchard, Stephan Rasp, Gael Reinaudi, and Galen Yacalis. Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters*, 45(11):5742–5751, 2018.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Neural Information Processing Systems*, 2014.

[24] Zekun Hao, Xun Huang, and Serge Belongie. Controllable video generation with sparse trajectories. In *Conference on Computer Vision and Pattern Recognition*, 2018.

[25] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Neural Information Processing Systems*, 2017.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[27] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *European Conference on Computer Vision*, 2018.

[28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

[29] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In *Neural Information Processing Systems*, 2015.

[30] Yunseok Jang, Gunhee Kim, and Yale Song. Video prediction with appearance and motion conditions. In *International Conference on Machine Learning*, 2018.

[31] Dinesh Jayaraman, Frederik Ebert, Alexei A Efros, and Sergey Levine. Time-agnostic prediction: Predicting predictable video frames. In *International Conference on Learning Representations*, 2019.

[32] Nal Kalchbrenner, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *International Conference on Machine Learning*, 2017.

[33] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[34] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *Conference on Computer Vision and Pattern Recognition*, 2019.

[35] 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. *arXiv:1705.06950*, 2017.

[36] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[37] Adam R. Kosiorek, Hyunjik Kim, Ingmar Posner, and Yee Whye Teh. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Neural Information Processing Systems*, 2018.

[38] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. In *Neural Information Processing Systems*, 2019.

[39] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv:1804.01523*, 2018.

[40] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Flow-grounded spatial-temporal video prediction from still images. In *European Conference on Computer Vision*, 2018.

[41] Yitong Li, Martin Renqiang Min, Dinghan Shen, David Carlson, and Lawrence Carin. Video generation from text. In *AAAI Conference on Artificial Intelligence*, 2018.

[42] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *arXiv:1705.02894*, 2017.

[43] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection – a new baseline. In *Conference on Computer Vision and Pattern Recognition*, 2018.

[44] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *International Conference on Computer Vision*, 2017.

[45] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. In *International Conference on Computer Vision*, 2017.

[46] Pauline Luc, Camille Couprie, Yann LeCun, and Jakob Verbeek. Predicting future instance segmentation by forecasting convolutional features. In *European Conference on Computer Vision*, 2018.

[47] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *International Conference on Learning Representations*, 2016.

[48] Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. In *International Conference on Learning Representations*, 2018.

[49] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[50] Thanh Thi Nguyen, Cuong M Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen, and Saeid Nahavandi. Deep learning for deepfakes creation and detection. *arXiv preprint arXiv:1909.11573*, 2019.

[51] M. Oliu, J. Selva, and S. Escalera. Folded recurrent neural networks for future video prediction. In *European Conference on Computer Vision*, 2018.

[52] Viorica Pătrăucean, Ankur Handa, and Roberto Cipolla. Spatio-temporal video autoencoder with differentiable memory. In *International Conference on Learning Representations*, 2016.

[53] Marc'Aurelio Ranzato, Arthur Szlam, Joan Bruna, Michaël Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv:1412.6604*, 2014.

[54] Fitsum A. Reda, Guilin Liu, Kevin J. Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao, and Bryan Catanzaro. Sdc-net: Video prediction using spatially-displaced convolution. In *European Conference on Computer Vision*, 2018.

[55] David Rolnick, Priya L Donti, Lynn H Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, et al. Tackling climate change with machine learning. *arXiv preprint arXiv:1906.05433*, 2019.

[56] Ekraam Sabir, Jiaxin Cheng, Ayush Jaiswal, Wael AbdAlmageed, Iacopo Masi, and Prem Natarajan. Recurrent convolutional strategies for face manipulation detection in videos. *Interfaces (GUI)*, 3:1, 2019.

[57] Masaki Saito and Shunta Saito. TGANv2: Efficient training of large models for video generation with multiple subsampling layers. *arXiv:1811.09245*, 2018.

[58] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014.

[59] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Neural Information Processing Systems*, 2015.

[60] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Deep learning for precipitation nowcasting: a benchmark and a new model. In *Neural Information Processing Systems*, 2017.

[61] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012.

[62] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *International Conference on Machine Learning*, 2015.

[63] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing motion and content for video generation. In *Conference on Computer Vision and Pattern Recognition*, 2018.

[64] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv:1812.01717*, 2018.

[65] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *International Conference on Learning Representations*, 2017.

[66] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. In *International Conference on Machine Learning*, 2017.

[67] C. Vondrick, P. Hamed, and A. Torralba. Anticipating the future by watching unlabeled video. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[68] Carl Vondrick and Antonio Torralba. Generating the future with adversarial transformers. In *Conference on Computer Vision and Pattern Recognition*, 2017.

[69] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Neural Information Processing Systems*, 2016.

[70] Niklas Wahlström, Thomas B. Schön, and Marc Peter Deisenroth. From pixels to torques: Policy learning with deep dynamical models. In *ICML Deep Learning Workshop*, 2015.

[71] Jacob Walker, Abhinav Gupta, and Martial Hebert. Dense optical flow prediction from a static image. In *International Conference on Computer Vision*, 2015.

[72] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, 2016.

[73] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612, 2004.

[74] Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. A locally linear latent dynamics model for control from raw images. In *NeurIPS*, 2015.

[75] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. In *International Conference on Learning Representations*, 2020.

[76] Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, and Jiebo Luo. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In *Conference on Computer Vision and Pattern Recognition*, 2018.

[77] Jingwei Xu, Bingbing Ni, Zefan Li, Shuo Cheng, and Xiaokang Yang. Structure preserving video prediction. In *Conference on Computer Vision and Pattern Recognition*, 2018.

[78] Xinsheng Xuan, Bo Peng, Wei Wang, and Jing Dong. On the generalization of gan image forensics. In *Chinese Conference on Biometric Recognition*, pages 134–141. Springer, 2019.

[79] Tianfan Xue, Jiajun Wu, Katherine L Bouman, and William T Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Neural Information Processing Systems*, 2016.

[80] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *International Conference on Computer Vision*, 2017.

[81] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, 2019.

[82] Guangxiang Zhu, Zhiao Huang, and Chongjie Zhang. Object-oriented dynamics predictor. In *Neural Information Processing Systems*, 2018.

A Overview of DVD-GAN-FP

Except for the improvements discussed in the main text, our model is identical to DVD-GAN-FP [12]. This model consists of an encoder $\mathcal{E}$, a generator $\mathcal{G}$ and a discriminator $\mathcal{D}$. We give an overview of the architecture of each component here, and refer the reader to the paper for additional details. Both BigGAN and DVD-GAN were initially formulated in a class-conditioned setting. For simplicity, DVD-GAN-FP replaces the class label with a dummy constant zero class label wherever a class label is used.

$\mathcal{G}$ is primarily a 2D convolutional residual network based on the BigGAN architecture [6], with convolutional recurrent units interspersed between blocks at multiple resolutions to model consistency between frames. First, a feature vector is obtained by concatenating a latent sample z drawn from a normal distribution and features linearly obtained from the class label, both of dimension 128. Spatial features of resolution 8×8 are obtained via a reshaped affine transformation of this feature vector. These features are repeated along the time dimension, and passed for each time-step as input to the first convolutional recurrent unit. The features output at each time step then undergo framewise processing: a convolutional layer, followed by two identical BigGAN G residual blocks, the second of which performs nearest-neighbour spatial upsampling by 2. The recurrent unit and the framewise processing form the first stage of the architecture. Another two similar stages are employed, leading to features of resolution 64×64. At each stage, the recurrent unit's hidden state is initialized with the features extracted by the encoder at that resolution. Finally, the predicted images are obtained through a final BatchNorm-ReLU, a convolutional layer and a tanh. Batch normalization layers are also employed in each of the G residual blocks, independently for each time step, and conditioned on the 256-dimensional input feature vector containing the latent sample and the class information. All linear and convolutional layers in the entire model have Spectral Normalization applied to their weights [81], with the exception of the linear embedding mapping to the first spatial features.

All discriminators discussed in the main text are convolutional networks almost identical to BigGAN's discriminator, except in the input they take. Per-frame discriminators are entirely unchanged: they consist of five stages, each containing a single D residual block, the first four of which employ average pooling to downsample the features by 2. The final block is followed by a ReLU activation. Features are then summed across the spatial dimensions, and passed to the projection head, further described in Section E.1. The spatio-temporal discriminators, which are responsible for assessing temporal consistency, process the inputs identically, with the exception that all convolutions in the first two residual blocks are replaced with 3D convolutions.

In order to condition $\mathcal{G}$ on initial frames, an encoder $\mathcal{E}$, identical to the spatial discriminator network, is applied to the conditioning frames. For each RNN present in $\mathcal{G}$, the features extracted by the encoder at the corresponding resolution are reshaped, by folding the time dimension inside the channel dimension. The result is compressed using a single convolutional layer, to form the initial state for the corresponding RNN. In this way, all the recurrent units in $\mathcal{G}$ are initialized with features that have knowledge of the conditioning frames.

We refer the reader to [12] for the parametrization of the G and D residual blocks, and the corresponding hyperparameters, which our method leaves unchanged, unless stated otherwise.

B Kernel-based warping

We give a more formal description of factorized and pixelwise kernel-based warping, as introduced in Section 2.3. Pixelwise kernel-based warping approaches employ a network f to predict a set of parameters $\theta = \mathbf{W} \in \mathbf{R}^{H \times W \times k^2}$, given input x. These parameters are then used in depthwise, dynamic, locally-connected layers of kernel size k, taking as input $h \in \mathbf{R}^{H \times W \times C}$. In previous kernel-based video prediction approaches [79, 20, 68, 54], h and x are respectively the last conditioning frame and the concatenated conditioning frames. In the case of our recurrent units, h represents the

past hidden state h_{t-1} and the input x is formed by the concatenation of h_{t-1} and the current input to the unit x_t. Formally, at a given spatial position (i, j), the c^{th} dimension of the output vector $\tilde{h}_{i,j}$ is given by:

$$\tilde{h}_{i,j}[c] = \sum_{(m,n)\in[\![0,k-1]\!]^2} \mathbf{W}_{i,j}[mk+n] \cdot h_{i+m-(k-1)//2, j+n-(k-1)//2}[c], \tag{11}$$

where h has been padded to preserve spatial dimensions.

In the case of factorized kernel-based warping, f predicts $\theta = (\mathbf{w}, \mathbf{S})$, where $\mathbf{w} \in \mathbf{R}^{k^2 \times N}$ and $\mathbf{S} \in \mathbf{R}^{H \times W \times N}$, which are combined to form a tensor $\mathbf{W} \in \mathbf{R}^{H \times W \times k^2}$ as follows. At (i, j), the q^{th} dimension of $\mathbf{W}_{i,j}$ is given by :

$$\mathbf{W_{i,j}}[q] = \sum_{l=1}^{N} \mathbf{S}_{i,j}[l] \cdot \mathbf{w}[q, l]. \tag{12}$$

$\mathbf{W}$ is then used as in the pixelwise warping case. We illustrate the procedure for pixelwise and factorized kernel-based warping in Figure 4.

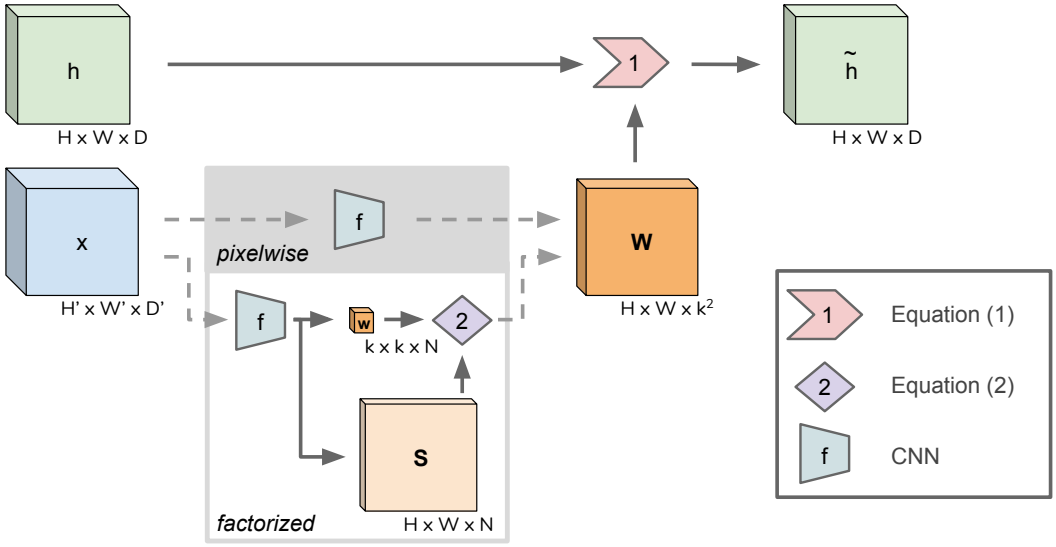

Figure 4: Kernel-based warping. Given input x, a network f predicts transformation parameters: a tensor of weights $\mathbf{W}$ for pixelwise warping, or a set of weights $\mathbf{w}$ and a map of coefficients $\mathbf{S}$ for factorized warping, which are then combined to obtain $\mathbf{W}$ (Equation (12)). In both cases, the result $\mathbf{W}$ is used in a dynamic, locally connected, depth-wise layer, applied to h (Equation (11)).

C Additional experiments

C.1 BAIR Robot Pushing

The BAIR Robot Pushing dataset of robotic arm motion [18] has been used as a benchmark in prior work on video prediction [2, 14, 39, 64, 12, 75]. The original BAIR Robot Pushing dataset contains 43,264 training samples and 256 test samples, each of 30 frames. Unterthiner et al. [64] have shown that the expected FVD between two non-overlapping 256-sized subsets of 16-frame videos randomly drawn from this dataset is on the order of 10^2. This suggests that on such a small test set, this metric has a very important bias, impeding quantitative comparison to state-of-the-art results. Instead, if we take two non-overlapping 30k-sized subsets, the expected FVD falls close to zero [64]. We therefore wish to employ 30k samples to estimate the statistics of both ground truth and generated samples. To do so, we propose to take a more balanced split of the data, using only the first 70% of the train set for training, and the remaining 30% augmented with the original 256 test samples for evaluation.

Table 4: FVD on BAIR Video Prediction, using the original (O) and balanced (B) splits of the BAIR dataset. On the original split, FVD is estimated using 256 samples, due to the small test set size, leading to a large bias for the FVD (on the order of 100.) We propose a more balanced split of the data, to allow the use of 30K samples to estimate the test FVD.

METHOD	SPLIT	FVD (↓)
SAVP	O	116.4
DVD-GAN-FP	O	109.8
TRIVD-GAN-FP	O	103.3
VIDEO TRANSFORMER	O	**94** $\pm$ **2**
TRIVD-GAN-FP	B	**31.8** $\pm$ **0.2**

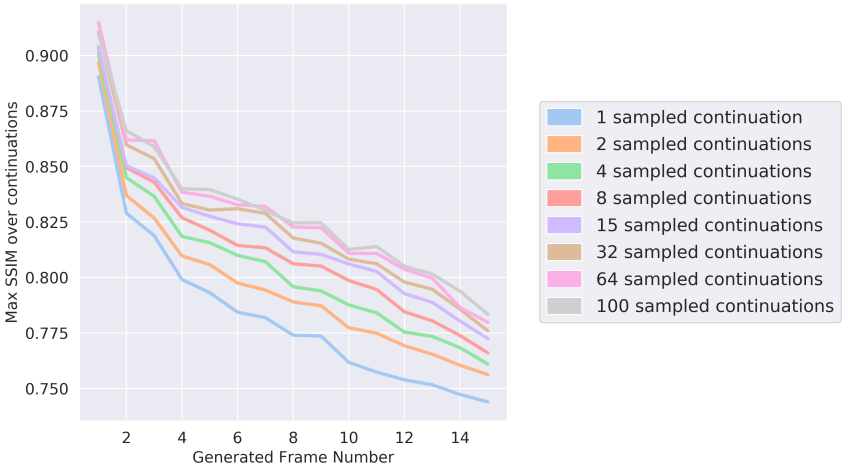

Figure 5: Per-frame SSIM for TrIVD-GAN-FP where SSIM is taken as the max over differing numbers of sampled continuations. Higher is better.

We train TrIVD-GAN-FP, using a channel multiplier in $\mathcal{G}$ of 48, and in $\mathcal{D}$ of 92, and learning rates of $2 \cdot 10^{-5}$ for $\mathcal{G}$ and $3 \cdot 10^{-5}$ for $\mathcal{D}$. Like previous work, we condition on one frame and generate 15 future frames. As on Kinetics, we validate models according to performance on the train set and report FVD on the test set for both splits. We use the original split to compare to Video Transformer and DVD-GAN-FP, and our proposed balanced split, as a baseline for future work.

We report the obtained results on this dataset in Table 4. Despite substantial improvement over DVD-GAN-FP and Video Transformer [75] on Kinetics, on BAIR, TrIVD-GAN-FP only sees slight improvement over DVD-GAN-FP, still higher than Video Transformer. Qualitatively, we do not observe a visual difference between the models, and echo comments from both papers that FVD may not be an adequate metric on BAIR, especially given the large bias of the metric resulting from the small sample size. Given access to 30K samples to estimate the FVD with the more balanced split, our model reaches 31.8 test FVD, estimated over 10 evaluation runs.

Finally, we report SSIM scores from BAIR, collected as in SAVP [39]. This involves generating ℓ continuations of a single frame and selecting the one which maximizes the average frame SSIM, then reporting per-frame SSIMs for all "best continuations" of conditioning frames. The results of this experiment are in Figure 5. Notably, we see high correlation between SSIM and increasing ℓ, meaning that generating more samples leads to a meaningfully diverse variety of futures, such that some are closer to the ground truth than others.

C.2 UCF-101

UCF-101 [61] is dataset of 13,320 videos of human actions across 101 classes. As a complement to the qualitative results we present, we also provide quantitative performance on this dataset. For the models with a channel multiplier of 48 trained for 1M iterations on Kinetics, TrIVD-GAN-FP improves upon DVD-GAN-FP from 117 test FVD to 95.8. The large model with a channel multiplier of 120 trained on Kinetics further improves this to 67.7. Finally, we train a model with channel multiplier of 92 on the UCF train set for 1M iterations, using the same hyperparameters as on Kinetics, and obtain our best results, of 55.2 ± 0.64 test FVD.

D Extended qualitative analysis

We start by showing in Section D.1 a qualitative comparison between recurrent units, using our medium-size models, described in Section 4.3 of the main paper. Next, in Sections D.2 and D.3, we provide further qualitative analysis of the samples predicted by our final large-scale TrIVD-GAN-FP model, described in Section 4.4 of the main paper. To support our analysis, we provide accompanying videos at https://drive.google.com/open?id=1fvmEm3gOWprWy2IuMFe4DuwEPahe9MwC.

All models have been trained on Kinetics-600 train set, and all predictions have been sampled randomly on the UCF-101 test set 1, due to restrictions on Kinetics-600.

D.1 Qualitative comparison between ConvGRU and TSRU

We now compare our proposed recurrent unit TSRU_p with ConvGRU, using for each the best model obtained when trained for 1M steps. We employ the truncation trick [6], as we find that this significantly improves both models' predictions. Specifically, we employ moderate truncation of the latent representation, using threshold 0.8.

We refer the reader to the accompanying video in folder *C.1.*, showing a side-by-side comparison of randomly sampled predictions, with the ConvGRU baseline on the left, and the TSRU model on the right. We find that our proposed module yields a slight but perceptible improvement over ConvGRU, in that the model using TSRU is more often able to conserve a plausible object shape across frame predictions.

D.2 Influence of the latent representation on the predictions

Here, we provide a qualitative analysis of the influence of the latent representation on the predictions, using our large-scale TriVD-GAN-FP model. In the directory *C.2.*, we provide a comparison of samples obtained using, on the left constant zero latent representations z (i.e. extreme truncation) and on the right random latent representations sampled from the same distribution as the one used during training (i.e. no truncation).

We find that in comparison with the constant zero latent representation samples, the non-truncated samples exhibit much more drastic changes in the predictions with respect to the input frames. Camera motion, zooming or object motion tend to be more important in a number of samples when using the non-truncated distribution. Effects like fade-to-black or novel object appearances, tend to appear regularly in the non-truncated examples, while we have not observed them in the samples that use a fixed zero latent representation. We point out examples of such effects in the annotated comparison (*annotated_constantz_vs_notruncation*); and also provide a non-annotated comparison (*constantz_vs_notruncation*). This points to a significant influence of the random variable on the predictions, as also evaluated quantitatively on the BAIR dataset in Section C.1.

Finally, for the interested reader, we also provide a larger batch of random samples in folder *C.2./all*, with the following levels of truncation: extreme truncation, where we sample all videos with the fixed zero latent representation; moderate truncation with 0.8 threshold; and no truncation at all.

D.3 Analysis of success and failure cases

As a result of the findings described in Section D.2, we find that our large-scale model also benefits from the truncation trick, in terms of generation quality. Focusing on the constant zero latent

representation samples, overall, we find that our final model TRiVD-GAN-FP generates highly plausible motion patterns, including in highly complex cases such as intricate hand motions or water motion patterns. The model also makes plausible predictions of complex motion in a number of cases, where points move with a non-uniform acceleration across frames, suggesting that it has learned a good representation for physical phenomena such as gravity (eg. for people jumping or juggling) and for some forms of object interactions (eg. for demonstrations of knitting). We highlight some of these in the accompanying video *C.3./annotated_success_constantz*.

We also highlight some failure cases in *C.3./annotated_failure_constantz_vs_notruncation*, where we provide constant zero latent representation samples on the left, and non-truncated samples on the right. We find that when the truncation trick is not used, the model sometimes predicts very large motion, that leads to important object deformations. We also sometimes observe large artifacts across the entire scene. Occasionally, the model makes predictions of obviously implausible trajectories. In contrast, when using truncation, we see that these failures cases largely disappear.

Across both levels of truncation however, we find that the model struggles when globally coherent motion must be predicted for objects with fine spatial structure, eg. in the case of hoola loops, or horse legs. We also find that in certain cases, motion seems to simply stop, presumably since this is an easy way for the model to yield plausible predictions in an overly conservative manner. We note however that a trivial *copy* baseline, which outputs a copy of the last input frame for each time step, performs very poorly at 330 on the Kinetics validation set - showing that consistently predicting trivial motion patterns is heavily penalized by the FVD. On rare occasions, we also observe objects disappearing into the void. A final failure case comes up when the input sequence is blurry, which leads to poor predictions.

E Experimental Details

E.1 Loss

The discriminator and generator are jointly trained with a geometric hinge loss [42], updating the discriminators two times for each generator update. Specifically, the discriminator optimizes the following empirical risk:

$$\frac{1}{D}\sum_{d=1}^{D}\rho(1+\mathcal{D}(\mathcal{G}(z)))+\rho(1-\mathcal{D}(x)), \tag{13}$$

where D is the discriminator output dimension, ρ is the ReLU function, and z and x represent respectively a latent sample and a video. The generator optimizes the following empirical risk:

$$-\frac{1}{D}\sum_{d=1}^{D}D(G(z)). \tag{14}$$

Following DVD-GAN, we employ a variant of the projection conditioning of the discriminator [2], which we call *mixed projection discriminator*.

We first recall the formulation of the original projection discriminator [48]. Calling h a batch of spatio-temporal features extracted from the videos fed to the discriminator, the projection discriminator maps these, on the one hand, to per-sample r_x^b scalars only dependent on the input videos. On the other, a dot product is taken between h and a learned vector, corresponding to a dummy zero class y (or to the conditioning class if appropriate), to obtain a per-sample $r_{x,y}^b$ scalar dependent on both the videos and the class. Next, these scalars are added together to form the discriminator output o_b, consisting of a single scalar per sample b.

Instead, the mixed projection discriminator extracts scalars $r_x^{b,t}$ and $r_{x,y}^{b,t}$ for each batch sample b and each time step t and computes $D = B \times B \times T$ outputs, where B represents the batch size and T the time dimension of h, obtained as follows:

[2] Communicated to us privately by the authors to help reproduce their results.

$$o_{b,b',t} = r_x^{b,t} + \sum_{t'} r_{x,y}^{b',t'}. \tag{15}$$

We find that this variant improves over the original projection discriminator from 64.2 to 55.0 on the Kinetics validation set, for medium-sized models using a channel multiplier of 48 trained for 700k steps. As a result we employ it to train all our models.

E.2 Implementation Details

We preprocess the data as follows. Real video clips are first resized so that their smallest dimension is 64, then randomly cropped and finally scaled to the range [-1, 1]. For evaluation, real and generated videos are then resized to resolution 224×224 using bilinear interpolation, for input to the I3D network.

Spectral Normalization [81] is applied to all weights in the generator, approximated by only the first singular value. All weights are initialized orthogonally [58], and during training we maintain an auxiliary loss on the generator which penalizes weights diverging from orthogonality. This loss is added to the GAN loss before optimization with weight 0.0001. The model is optimized using Adam [36] with batch size 512, β_1 set to 0, β_2 set to 0.999 and a learning rate of $1 \cdot 10^{-4}$ and $5 \cdot 10^{-4}$ for $\mathcal{G}$ and $\mathcal{D}$ respectively.

All samples for evaluation (but not for training) use an exponential moving average of $\mathcal{G}$'s weights, which is accumulated with decay $\gamma = 0.9999$ and is initialized after 20,000 training steps with the value of weights at that point. In order to disambiguate the effect of the weights' moving averages from those present in the Batch Normalization layers, we always evaluate using "standing statistics", where we run a large number of forward passes of the model (100 batches of 256 samples), and record the average means and variances for all Batch Normalization layers. These averages are used as the "batch statistics" for all subsequent evaluations. This process is repeated for each evaluation.

Conditioning of the Batch Normalization layers [28, 13, 17] is done by having the scale and offset parameters be an affine transformation of the conditioning vector. We note that the final batch normalization layer in $\mathcal{G}$ is not conditional, and uses a regular learned scale and offset.

E.3 Estimating average step duration and memory consumption

We note that for calculating step time and maximum memory usage in Table 1 of the main paper, each training step corresponds to two $\mathcal{D}$ updates and a single $\mathcal{G}$ update, involving three passes of $\mathcal{G}$ with batch size 8 and two passes of $\mathcal{D}$ with batch size 16 alongside one with batch size 8. Furthermore, we emphasize that TPUs rely on the XLA intermediate compilation language, which automatically performs some re-materialization and optimization. As such, these numbers might change for other implementations.

F Further details on the recurrent units

F.1 Implementation details

For the ConvGRU modules in DVD-GAN-FP, Clark et al. [12] use ReLU activation functions, rather than the hyperbolic tangent (tanh). For the ConvLSTM module, we experiment with both tanh, which is traditionally used, and the ReLU activation. We find that ReLU significantly outperforms tanh early on, reaching 51.8 train FVD after 180K steps, against 62.7 for tanh. As a result, for our comparison of recurrent units, we report results using the ReLU activation function.

The hyper-network f used to predict the parameters of our transformations differs depending on the output dimension. In the case of PTSRU, we use a resolution-preserving, 2-hidden layer CNN. For both TSRU and K-TrajGRU, a first subnetwork consists of a single convolutional layer, followed by adaptive max-pooling, of output resolution 4×4, then by a single hidden-layer MLP, to predict the set of kernels $\mathbf{w}$ used across the input. The pseudo-segmentation map $\mathbf{S}$ is obtained using a single convolutional layer applied to the input. For all three architectures, ReLU activations are interleaved between the linear layers, and we use a softmax activation on the output. Each architecture has zero

or two hidden layers, and we set the number of output hidden channels (or units in case of linear layers) to half the number of input channels.

F.2 K-TrajGRU

Our proposed K-TrajGRU is a kernel-based extension of TrajGRU [60], which we further detail here. Formally, it is described by the following equations:

$$\theta_{h,x} = f(h_{t-1}, x_t), \tag{16}$$
$$\tilde{h}_{t-1} = w(h_{t-1}; \theta_{h,x}), \tag{17}$$
$$r = \sigma(W_r \star_k [\tilde{h}_{t-1}; x_t] + b_r), \tag{18}$$
$$h'_t = r \odot \tilde{h}_{t-1}, \tag{19}$$
$$c = \rho(W_c \star_k [\tilde{h}'_t; x_t] + b_c), \tag{20}$$
$$u = \sigma(W_u \star_k [\tilde{h}_{t-1}; x_t] + b_u), \tag{21}$$
$$h_t = u \odot h_{t-1} + (1 - u) \odot c, \tag{22}$$

where h_{t-1} and x_t denote respectively the past hidden features and the input features; f and w are respectively the shallow convolutional neural network and the factorized warping function presented in the main paper and used by our TSRU modules; $\star_k$ denotes a convolution operation with kernel size k, ρ is the activation function used (in our case, tanh), σ denotes the sigmoid function. $\odot$ denotes elementwise multiplication, and W_o and b_o are kernel and bias parameters for the convolutions used to produce each intermediate output o.

Conceptually, we see that K-TrajGRU and TSRU are related, in that they both warp the past hidden features to account for motion. TSRU however has a simpler design, which can be intuitively interpreted: it fuses the warped features $\tilde{h}_{t-1}$ and novelly generated content c, using a predicted gate u. K-TrajGRU, instead, provides the warped features to equations (18, 20, 21), but still combines the input hidden features h_{t-1} unchanged, together with c.

We note that Shi et al. [60] propose to predict L warped versions of the hidden state, and that our proposed recurrent unit specifically extends the case where $L = 1$. Increasing L, as well as the addition of a read gate to the TSRU module, are potential ideas for further improving our transformation-based recurrent unit's expressivity, which we leave to be explored in future work.

F.3 ConvLSTM

For comparison with other recurrent units, the hidden state and memory cell of our ConvLSTM baseline each have half as many channels as the hidden state used in the other designs, that do not maintain a memory cell (ConvGRU, K-TrajGRU and TSRU variants). The hidden state and the memory cell are initialized with respectively the first and the second half of the conditioning sequence's encoding. Like for the other recurrent units, this encoding is obtained by passing the concatenated image-level features extracted by the encoder for each frame, through a 3x3 convolution layer, followed by a ReLU activation, to compress the representation to the right dimension. Then, the output hidden state and memory cell for each time step are concatenated and passed on to the subsequent residual blocks of the generator.

F.4 Dynamic ConvLSTM

The dynamic ConvLSTM proposed by Xu et al. [77] is conceptually simple: for each sample, a set of kernels are predicted and used in the convolutional layers applied either on the recurrent unit's input features x or on its hidden features h, in the input, forget, output gate equations, as well as for the intermediate state's equation - totaling 8 predicted kernels. The network predicting these kernels, denoted by "Kernel CNN", employs a channel-wise softmax operation along the input channel to increase sparsity of the predicted kernels. We refer the reader to the Section 3.3 of the original paper for a more extensive presentation.

It is important to note that such dynamic convolutions require a different $2D$ kernel to be predicted for each input and output channel. To keep to a small number of parameters of the Kernel CNN, Xu et al. [77] propose to share weights across the input and output channel dimensions, by mapping the difference between image features extracted from the two previous frames $F_t[i] - F_{t-1}[j]$ to the corresponding 2D kernel $W(i, j)$, where $F[i]$ denotes the $i - th$ channel of features F, and where $W(i, j)$ corresponds to the filter used for input channel i and output channel j. This relies on the fact that the generator is autoregressive, and encodes each prediction before making the next one. We adapt this to our architecture, as described later in this section for the sake of completeness.

Before going into more implementation details however, we draw the attention of the reader to the following important analysis. In contrast with dynamic convolutions, our proposed transformation-based recurrent units all essentially rely on dynamic, depth-wise, locally connected layers. Hence, while these recurrent units predict transformation parameters whose dimension scales with the spatial dimensions of the hidden features, the dimension of the Dynamic ConvLSTM transformation's parameters scales with the number of input and output channels. We find that this leads to a largely increased computational cost, even for medium sized models. Specifically, with extensive use of parallelisation across devices, when using Dynamic ConvLSTMs, our medium-size models, that use a channel multiplier of 48, run at an average step duration of 6.385s for a batch size of 512; while in the same set up, models using ConvGRU (resp. TSRU) run at 284ms (resp. 354ms) per step. In this context, the Dynamic ConvLSTM module hence only allows a very limited width of architectures. In this sense, we argue that it is not scalable, and therefore we do not consider it for comparison with other recurrent units for the purposes of our large-scale setting.

Implementation details To allow the use of their proposed recurrent unit in our architecture, at time step t, we treat the first c_h channels of the previous and current features (x_{t-1}, x_t) as the input features to the Kernel CNN, where c_h denotes the number of channels of the hidden representation of the recurrent unit. We note that this requires the number of channels c_x of the inputs features x to be a multiple of c_h, so that each difference of features can be mapped to $q + 1$ kernels, where q is such that $c_x = q \times c_h$. For the first time step, we use the first c_h channels of the sequence encoding to play the role of the previous features x_{t-1}.

A second difference is that their proposed unit additionally fuses the past hidden features, and past cell states to produce the hidden and cell states that are input to the module. This operation is denoted by $Fus - 4$ in the paper. This might be redundant with the intended function of the hidden state and memory cell of a ConvLSTM. In practice, it is observed by the authors to bring only a marginal gain in general. Additionally, it is an orthogonal contribution, which could be implemented for any of the other recurrent units we have considered. As a result, we do not implement this aspect of their recurrent unit.

F.5 TSRU variants

We provide in Figure 6 an illustration for all proposed TSRU variants described in Section 3.2.

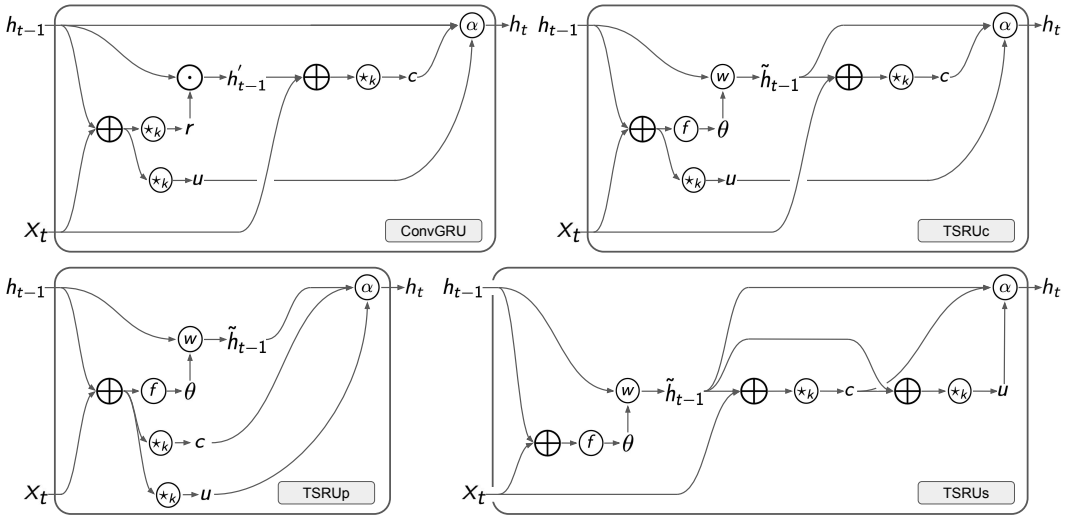

Figure 6: Information flow for ConvGRU (top-left) and our proposed TSRU$_c$ (top-right), TSRU$_p$ (bottom-left) and TSRU$_s$ variants (bottom-right); where α represents elementwise convex combination of h_{t-1} and c with coefficient provided by u; $\star_k$, convolution with kernel size k; $\bigoplus$, concatenation and $\bigodot$, elementwise multiplication.