Video prediction on MMNIST dataset

Mattia Castelmare
Dept. of Computer,
Control and Management Engineering
Antonio Ruberti

Andrea Giuseppe Di Francesco
Dept. of Computer,
Control and Management Engineering
Antonio Ruberti

Enrico Fazzi
Dept. of Computer,
Control and Management Engineering
Antonio Ruberti

***Abstract*—Forecasting a succession of frames, given a sequence of previous ones, is the core intuition behind the video prediction task.**
Over the years there have been developed several deep learning solutions to this problem, and still today novel architectures can push the current *state-of-the-art* (SOTA) to the limit. Even though deep learning approaches typically favor growth in the number of parameters to achieve better performances, the current trend in the video prediction task is the adoption of only classic architectures, such as convolutional neural networks (CNNs).
In this report, we explore the different aspects that should be considered to guarantee correct video forecasting, by replicating two of the latest works that contributed to the field. Both methods overcame solutions that implied sequence modeling architectures and reduced the overall complexity with novel temporal encoding alternatives. One of these is the current SOTA in various video prediction benchmark datasets.
The aforementioned methods are also compared with a simple baseline network that lacks the temporal information encoding between the input frames.

1. Introduction

In the literature, it has become clearer that achieving competitive results in the video prediction task entails a precise understanding of the spatiotemporal correlation in the input pixels ([1], [2], [3]). In our case, this understanding is meant to be automatically performed by a neural-based system.
According to [4], the pixels' spatial correlation is identified within the intra-frame features, whereas the temporal correlation is attributable to inter-frame features. Keeping this in mind, we understand why the video prediction task can be referred to as *Spatiotemporal predictive learning* (STPL) [4].
Improving the models in STPL underlines an important aspect, which is the possibility of producing higher-quality features for affine tasks to the one considered.
During the learning process, the model tries to learn a representation of the intrinsic motion dynamics so as to improve the feature extraction. In this sense the video prediction task can be seen as a pretext task, trained in a self-supervised fashion.
When the model has learned to predict, we would experience some emerging human-level behaviors, such as the ability to distinguish between the foreground and the background in a video.
In this report, we show how we replicated the work done in [5], and [4], with a particular concern on the Moving MNIST dataset, and how the temporal correlation representation is crucial to obtain outstanding performances. The experiments were reproduced at the best of our computational power and the knowledge provided in the available manuscripts, thus the results presented must not be taken into account for a general evaluation of the methodologies.

2. Architectures

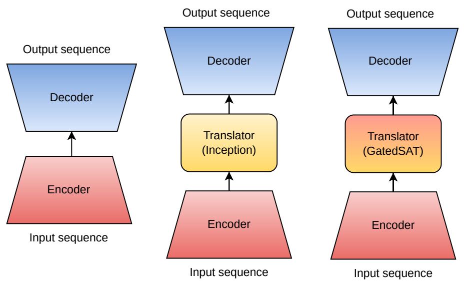

Figure 1. Video prediction Models. From left to right: Vanilla Encoder-Decoder, SimVP (inception) [5], SimVP (GatedSAT) [6].

2.1. Vanilla Encoder-Decoder

The first architecture implemented to solve the aforementioned task is a vanilla spatial Encoder-Decoder (VED) architecture. Striving for simplicity, the model follows the general framework shown in Figure 1, it is composed of a spatial Encoder with four convolutional 2D layers and a spatial Decoder of four 2D transposed convolutional layers. In order to preserve the spatial-dependant features we add a skip connection from the first convolutional layer in the Encoder to the last transposed one in the Decoder.
In our selection of the architecture, we took into consideration

the nature of the videos which consist of a sequence of consecutive frames. It is well-known that Convolutional Neural Networks (CNNs) are highly effective in the processing and analysis of image data.
In the encoding component of the architecture, more precisely in the downsampling step, a stride of two was applied; otherwise, we selected it equal to one. The same strategy was applied in the decoding stage, when upsampling was required. Each convolutional layer comprised a *Rectified linear unit* (ReLU) as activation function, and a Group Normalization layer [7]. Even though this choice was inspired by [5], Section 3.3 provides some evidence that supports this choice.

2.2. SimVP: Inception Module

The second model adopted is the Simpler yet Better Video Prediction (SimVP) [5] which consists of a spatial Encoder, a Translator module, and a spatial Decoder built on CNN as shown in Figure 2.
The novelty is the Translator: a module that is able to learn the video's temporal evolution, which is not taken into account in the VED. The combination of the spatial information captured by the Encoder and Decoder and the temporal information captured by the Translator enables the model to effectively extract valuable spatiotemporal information from videos.

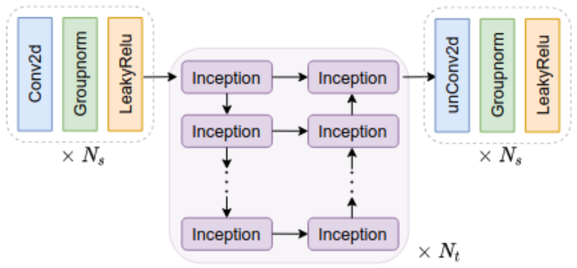

Figure 2. Inception module

The Translator employs N_t Inception modules to learn temporal information, each module as described in [5] is composed of a bottleneck Conv2d with 1x1 kernel followed by parallel GroupConv2d operators. Let us notice that this Inception module differs from the original introduced in [8]. The authors of [5] claim that they found this design more convenient for their architecture.
In the temporal module, the previous features encoded by the Inception Encoder are then passed to the next Inception Decoder concatenated with the previous decoded features as shown in 2.
In this architecture, the spatial Encoder and Decoder are composed of, respectively, Conv2d and transposed Conv2d layers, a Groupnorm, and a *LeakyRelu* [9] as an activation function.

2.3. SimVP: Gated SpatioTemporal Attention

Following the success of SimVP (Inception) [5], it was made clear that is possible to develop lightweight models to improve STPL. More recently, it has been released an even lighter version of the SimVP architecture, that substitutes the Inception-based translator, with an Attention unit based on large kernel convolutions.
Such configuration has shown to achieve competitive results on the task, with a lower number of parameters. The proposed method is introduced in [6], and we will refer to it as SimVP2 for simplicity.
In our experiments we have decided to compare VED and SimVP with the newest SimVP2. In this Section we report a brief description of the SimVP2 translator; later in the report, we illustrate its performances.
One of the novelties of this architecture is that it uses large kernel convolution so as to achieve a larger receptive field in the input processing. It follows, indeed, the same fashion of transformers and their attention mechanism.
The other novelty relies on the simplicity of the computation, which is lighter w.r.t. the inception modality proposed in SimVP.
A representation of the *Gated SpatioTemporal Attention* (GSTA) module is shown in Figure 3. A batch of input

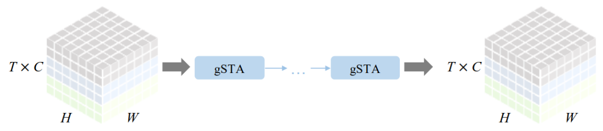

Figure 3. Encoder+Translator+Decoder in SimVP2.

frames can be described as $\mathcal{X} \in \mathbb{R}^{B\times T\times C\times H\times W}$, where T is the number of frames/sample, $C = 1$ for white & gray images, and H, W are both 64.
Once the input $\mathcal{X}$ is processed by the Encoder, we can reshape it to $B \times (T \cdot C') \times H' \times W'$, to discriminate the inter-frame features. In this case $C' = 64$ and $H', W' = 32$. Since the two translators have common functions, they also present the same number N_T of stacked layers. In the generic case, the translator should transform the input sequence to have a number of frames T equal to the number of predicted frames T'.
In the SimVP case $N_t = 3$, according to [5] hyperparameters' choices, whereas in SimVP2, $N_t = 8$. Apparently, this value seems to be set to $N_t = 8$, also in future SimVP experiments, but for our comparisons, we decided to keep the original setting as declared in [5].

2.3.1. GSTA description. Each GSTA module takes as input the Encoder output $\mathcal{H}$, and it returns the Decoder input $\mathcal{H}'$, that in our case shares the same dimension of $\mathcal{H}$, since $T = T' = 10$.
In the translator, modules are applied as described in the

following equations:

$$\begin{aligned}
&\hat{\mathcal{H}}' = Conv2d_{1\times1}(Conv_{Dw-d}(Conv_{Dw}(\mathcal{H}))),\\
&\bar{\mathcal{H}}'_a, \bar{\mathcal{H}}'_b = split(\hat{\mathcal{H}}'),\\
&\mathcal{H}' = \sigma(\bar{\mathcal{H}}'_a) \odot \bar{\mathcal{H}}'_b,
\end{aligned} \tag{1}$$

where $\sigma(\cdot)$ is the sigmoid function, that filters the most relevant features. For this reason, it is referred to as 'Gated'.

3. Implementation details

3.1. Dataset

We conducted our experiments on the Moving MNIST dataset: it contains 10,000 video sequences, each consisting of 20 frames. In each video sequence, two digits move independently in a 64 x 64 grid. Digits often overlap each other and bounce off the boundaries.

3.2. Loss function

The general loss function $\mathcal{L}_\theta$ is formalized as follows.

$$\mathcal{L}_\theta = \sum_{i=1}^{T'} ||\mathcal{Y}' - \mathcal{Y}||_2^2. \tag{2}$$

$\mathcal{Y}', \mathcal{Y}$ are the predicted and actual frames respectively.
The mean squared error (MSE) is a standard loss function, to assess the differences between two pixel-based signals. Other recent approaches were considered in recent research [4], such as the minimization of the Kullback-Liebler divergence in addition to the MSE, which, however, was not considered in [6].

3.3. Hyperparameter setting

The choice of hyperparameters was influenced by a trade-off between maximizing network performances and minimizing the overall number of parameters within it. We decided to train all the presented models for 100 epochs in order to have an overall idea of their behaviors.
In general, replicating SOTA's performances was not straightforward, since such results required a number of epochs higher than 1000 (e.g. [5]).
For all the training it was used the Adam optimizer [10] with a learning rate of 0.01, and an OneCycleLr scheduler [11]. For SimVP and SimVP2, we took inspiration from their original paper details. Their values are shown in Table 1.

lr	layers	kernel size	hidden dim
0.01	4	3 ×3	64

Table 1. COMMON HYPERPARAMETERS TO VED, SIMVP, SIMVP2

As regards our personal choices, we established the activation function and the normalization strategy in the convolutional layers. We adopted a ReLU activation function in VED experiments, and a LeakyReLU for the SimVPs architectures, according to [5], [6]. A Group normalization [7] layer was applied after each convolution both in VED and in the SimVPs models.
Before choosing the Group norm. layer, other normalization strategies were considered.
When using an architecture without any normalization there are no clear improvements over time (see Figure 4).
Instead, when using batch normalization [12] we experience better stability, but the high variance (different videos) at the batch level could lead to instability during the learning process. In successive trials we also tested the layer normalization [13] and the group normalization effects on the training. Our final choice was the latter since it has the most stable test loss over time (see Figure 4).

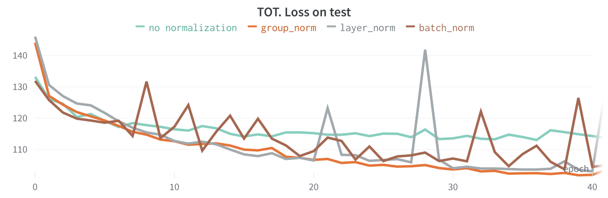

Figure 4. Comparisons between the different normalization layers during previous experiments that were replicating [4].

3.4. TAU module

Throughout our trials, we have tested also the Temporal Attention Unit (TAU) module [4]. The pivotal concept at the base of this block is the subdivision of the attention module into two parts: static attention (SA), which focuses on intra-frame dependencies, and dynamic attention (DA), which aims at catching inter-frame and hence temporal relations between frames.
Although this model was the SOTA in June 2022, we were unable to replicate the results shown in [4] as we lacked information such as details on the architecture and values of the hyperparameters. Our choice was to implement the most recent SOTA (July 2022) [6] in order to obtain better results than the SimVP architecture. However, in this report, we focus on the SimVP experiments, where we obtained better performance.

3.5. Results

3.5.1. Quantitative Results. In this section, we illustrate the results of all the architectures.
In figure 5 the VED test loss over time is represented. VED, without any temporal attention mechanism, is not able to effectively capture the dynamic properties of the video data. In qualitative results, we show how VED tends to learn to reproduce the input video, rather than predicting future frames.
In any case, this is also evidenced quantitatively, with the

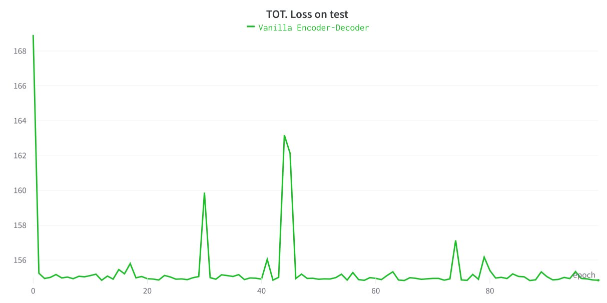

Figure 5. Loss on test Vanilla Encoder-Decoder

decreasing loss in Figure 5 that illustrates a plateau, which was immediately reached by the model in the first stages of the training phase.
As mentioned before, the second baseline tested for solving the task is the Simvp architecture.
The introduction of a temporal attention mechanism, in the form of a Translator module, led to an improvement in the performance of the Encoder-Decoder framework as shown in Figure 6. This is due to the ability of the Translator to extract temporal information from frames, effectively capturing the dynamic properties of the video data and allowing for more accurate prediction of future frames.

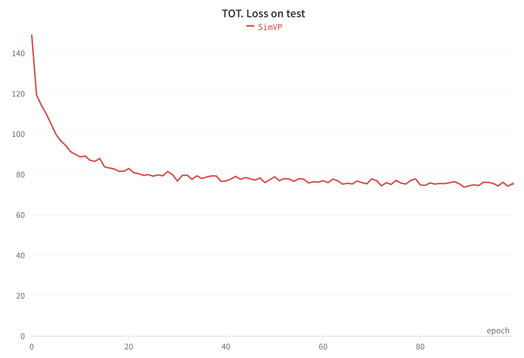

Figure 6. Training results on SimVP

The trend of the graph shows a decrease in the loss that reaches a value of 76 after about 100 epochs. We recall that in [5] the reported results were obtained after 2000 epochs. Compared to VED, Simvp introduces a significant improvement to the prediction task.
The training of the current SOTA SimVP2 module exhibits an even further improvement. In fact, as we can see in Figure 7, the MSE improves really fast, and in 100 epochs it is already possible to achieve competitive performances.
In Figure 8 we plot a comparison between the models. Here it is noticeable that the SimVP2 architecture outperforms the Inception-based SimVP, showing also a faster convergence.
In our opinion, this behavior is due to the model's intrinsic sequential nature. We observed that SimVP2 is more capable of understanding the spatiotemporal patterns in the videos

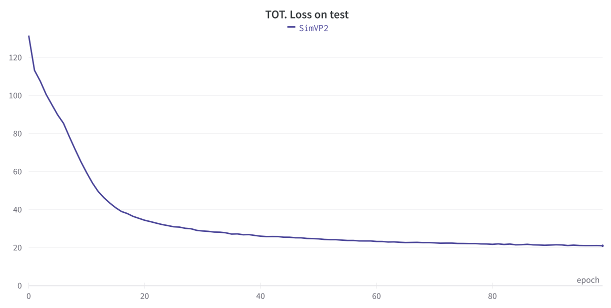

Figure 7. SimVP2: MSE loss on the test set.

with respect to SimVP. We think that this is attributable to the different architectures: the former is designed to resemble an attention network that processes the whole sequence of frames, the latter is built upon a UNet [14] structure and shows better flexibility in capturing spatial information rather than temporal one.
In Table 2 the results and the number of parameters for the configurations are listed.

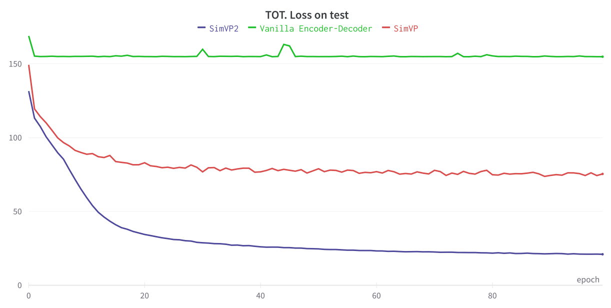

Figure 8. Comparison on test losses.

Model	MSE	n. of parameters
VED	154.839	0.297M
SimVP	75.428	23.4M
SimVP2	20.995	11.4M

Table 2. BENCHMARK OF THE MODELS. SIMVP2 ACHIEVES BETTER PERFORMANCES ON THE MOVING MNIST DATASET WHILE KEEPING THE LOWEST NUMBER OF PARAMETERS AMONG THE EXAMINED MODELS.

Qualitative results are shown in Figure 9.

4. Conclusions

In this report, we have defined three different architectures designed for the SPTL task.
Alongside a simple Vanilla Encoder-Decoder, we have tested two recently developed SOTA methods, SimVP (Inception), and SimVP (GSTA). Both methods outperform the Vanilla Encoder-Decoder, but SimVP (GSTA) reaches outstanding performances w.r.t. its Inception-based version, also exploiting a lighter-weight network size. The novelty of these works

lies in the CNN-based solutions. Despite the simplicity of the architectures, they are still able to achieve competitive results.
Future works may consider improving current performances just by means of simple adjustments to the architectures (e.g. more hyperparameters could be explored).
Moreover, increasing the complexity of the network could still be an alternative to push the SOTA further.
We firmly think that, even though CNNs outperformed more advanced architectures, this does not entail that solutions in other directions should not be attempted.

References

[1] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," 2015. [Online]. Available: https://arxiv.org/abs/1506.04214

[2] Y. Wang, Z. Gao, M. Long, J. Wang, and P. S. Yu, "Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning," 2018. [Online]. Available: https://arxiv.org/abs/1804.06300

[3] W. Yu, Y. Lu, S. Easterbrook, and S. Fidler, "Crevnet: Conditionally reversible video prediction," 2019. [Online]. Available: https://arxiv.org/abs/1910.11577

[4] C. Tan, Z. Gao, S. Li, Y. Xu, and S. Z. Li, "Temporal attention unit: Towards efficient spatiotemporal predictive learning," 2022. [Online]. Available: https://arxiv.org/abs/2206.12126

[5] Z. Gao, C. Tan, L. Wu, and S. Z. Li, "Simvp: Simpler yet better video prediction," 2022. [Online]. Available: https://arxiv.org/abs/2206.05099

[6] C. Tan, Z. Gao, and S. Z. Li, "Simvp: Towards simple yet powerful spatiotemporal predictive learning," 2022. [Online]. Available: https://arxiv.org/abs/2211.12509

[7] Y. Wu and K. He, "Group normalization," 2018. [Online]. Available: https://arxiv.org/abs/1803.08494

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014. [Online]. Available: https://arxiv.org/abs/1409.4842

[9] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015. [Online]. Available: https://arxiv.org/abs/1505.00853

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[11] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," 2017. [Online]. Available: https://arxiv.org/abs/1708.07120

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015. [Online]. Available: https://arxiv.org/abs/1502.03167

[13] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016. [Online]. Available: https://arxiv.org/abs/1607.06450

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

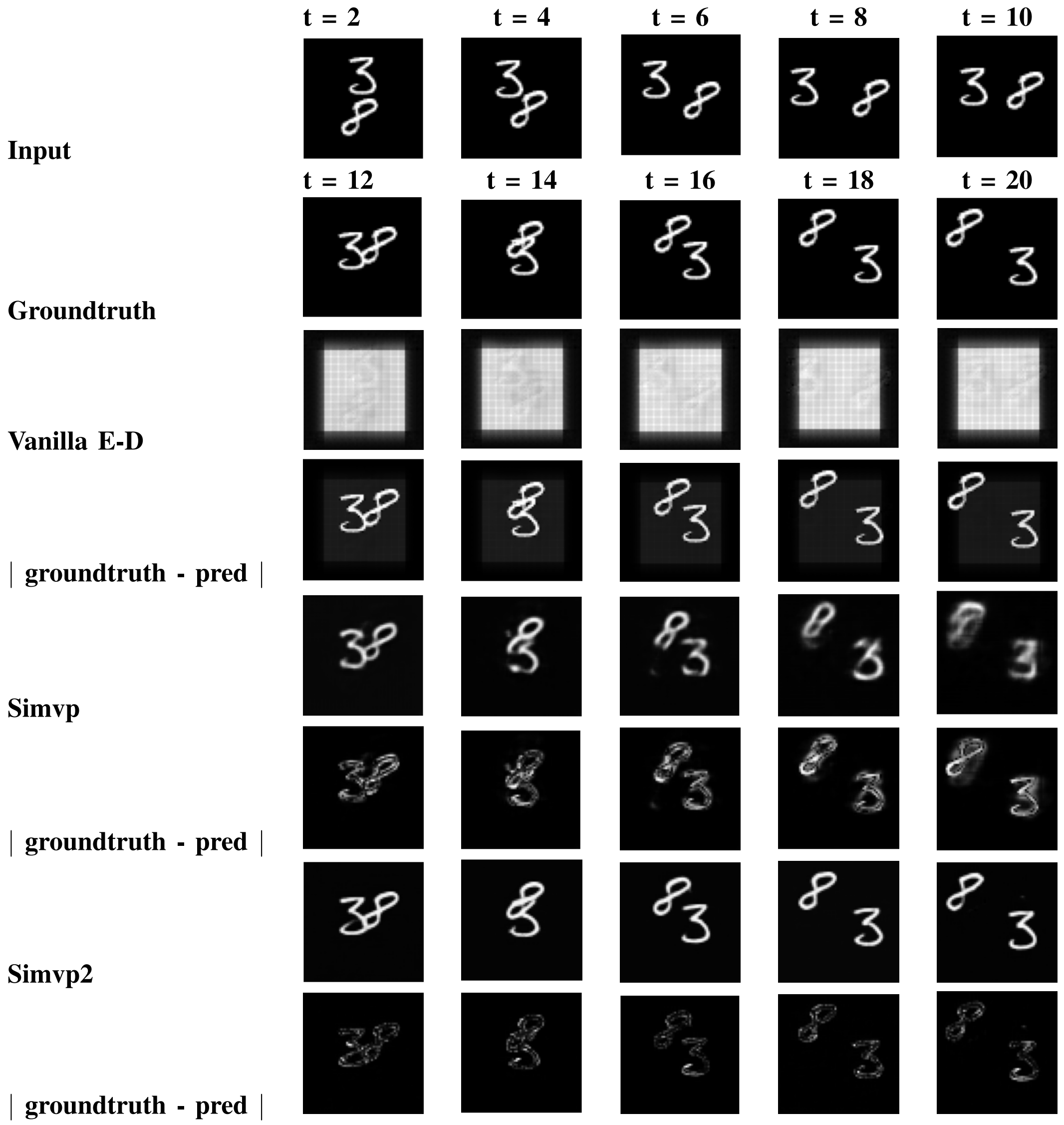

Figure 9. Plotting results