# Video Prediction on Moving MNIST

Gravina, Giovanbattista Nunziante, Luca

July 2023

**Abstract**

One of the emerging challenges in the application of Deep Learning in Computer Vision concerns making intelligent machines capable of predicting the future. To achieve this goal, research in spatiotemporal predictive learning has been on the rise for the past years, leading to solutions with more and more complexity both in terms of architecture and training strategies. In this work, a spatiotemporal predictive model made of convolutional layers only is analyzed, called SimVP. Despite the relatively simple structure and straightforward training approach, it is possible to train such model to obtain State of The Art performances. Experiments are carried out on the Moving MNIST dataset, comparing different models sharing the same chassis.

## 1 Introduction

It is easy to gauge the importance that anticipating the near future holds for any intelligent system. While humans do this effortlessly, it is not trivial to formalize and address this problem in an autonomous learning framework. Indeed, great interest has been shown towards this task, since it has many applications, ranging from vehicle prediction behaviour in autonomous driving to robot motion planning, but also weather nowcasting and safe human-robot interaction. Contributing to the complexity of such problem there are factors like lighting conditions variations, especially in outdoor scenarios, cluttered environments and occlusions.

The task of predicting future frames from a sequence of past frames is known as *Spatiotemporal predictive learning* since models need to extract meaningful spatial and temporal features. As the input data can be split to represent both past and future information, it is cast as a self-supervised learning problem. This report is structured as follows. In Section 2 the problem is formulated and some previous approaches are briefly presented. In Section 3 the used models are described. In Section 4 the loss function used and its various terms are explained. Finally, in Section 5 the experiments are detailed, and in Section 6 some conclusions are drawn.

## 2 Background

**Problem formulation** The problem at hand can be formalized as follows. Given a sequence of past $T$ frames at the current instant $t$ represented as a tensor $\mathcal{X}^{t,T} \in \mathbb{R}^{T \times C \times H \times W}$, the goal is to predict a sequence of future $T'$ frames $\mathcal{Y}^{t+1,T'} \in \mathbb{R}^{T' \times C \times H \times W}$ from time $t+1$, where $C, H, W$ are respectively the number of channels, height and width of the images.

**Previous works** Following [8], it is possible to divide the existing methods into four main categories:

- RNN-RNN-RNN: stack of multiple RNNs that process the input frames to make predictions (see Fig. 1a).
- CNN-RNN-CNN: process the input via a CNN encoder to extract features, use a RNN to predict future latent states and finally a CNN decoder decodes the latent space into the output frames (see Fig. 1b).
- CNN-ViT-CNN: same as the previous architecture, but using a Vision Transformer (ViT) to deal with the latent space (see Fig. 1c).

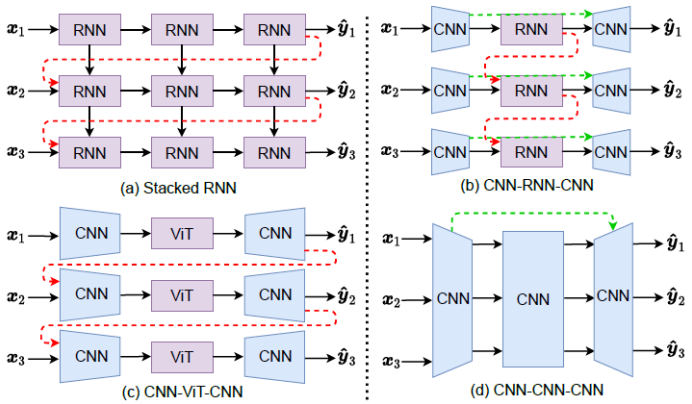



Figure 1: Major categories of the architectures for spatiotemporal predictive learning. The red and green dotted line are available to learn the temporal evolution and spatial dependency. Image and caption taken from [8].

- CNN-CNN-CNN: fully convolutional architecture performing predictions in one-shot, differently from all the previous methods which are sequential (see Fig. 1d).

This last approach is not as used as the others because, given the complexity of the task, it is thought that clever strategies must be applied to obtain State Of The Art performances. In contrast to this belief, in this work we analyze different architectures, all stemming from [8], that are fully convolutional and trained avoiding complex techniques, striving for simplicity.

# 3 Models

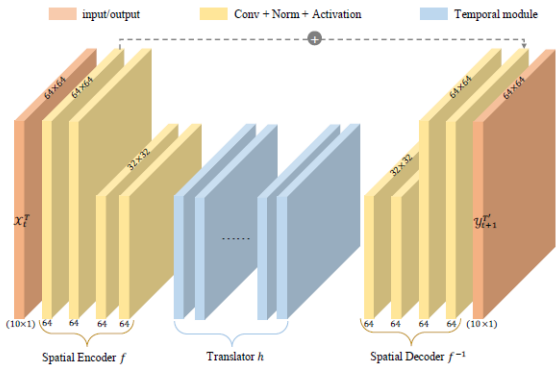



Figure 2: The overall SimVP architecture. The dimensions of the tensors are referred to the Moving MNIST dataset. Image taken from [8].

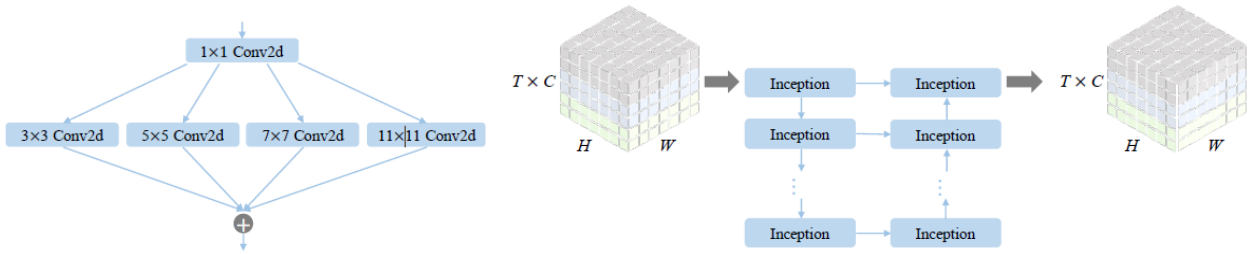
All the models employed in this work have the same autoencoder-like architecture of Fig. 1, called SimVP [8]. Differently from classical autoencoders, which reconstruct a single frame statically, SimVP encodes the past frames $\mathcal{X}^{t,T}$ and decodes the future frames $\mathcal{Y}^{t+1,T'}$.

The spatial encoder is made of $N_s$ vanilla convolutional layers and for every two layers downsampling is performed by setting the kernel stride to 2. It encodes the high dimensional past frames into the low-dimensional latent space, and treats each input frame as a single sample: indeed, given the input tensor shape $B \times T \times C \times H \times W$ where $B$ is the batch dimension, it is reshaped as $(B \times T) \times C \times H \times W$. To learn both spatial dependencies and temporal variations from sequential data in the translator, the hidden representations from the spatial encoder is reshaped by stacking multi-frame level features along the time axis, yielding a tensor $B \times (T \times C) \times H \times W$. The decoder, with $N_s$ convolutional layers and PixelShuffle layers to upsample the input every two convolutional layers, ultimately decodes the latent space into the

predicted future frames. The input to the spatial decoder is reshaped as $(B \times T) \times C \times H \times W$, and its output is reshaped to have the same shape as the input. As can be seen in Fig. 2, a residual connection is established between the first encoder layer and the last decoder layer to preserve spatial features.

In the following, three architectures are presented that use different spatiotemporal modules, one based on the Unet [6] structure and Inception modules [7], and two based on the general Metaformer abstraction [11].

## 3.1 Inception Unet




(a) Inception module (b) Inception Unet translator

Figure 3: In (a), the output of the different kernel sized convolutional layers are added up. Differently from the original Unet implementation, in (b) no contraction in the top-down path and no expansion in the bottom-up path are performed.

The first examined architecture was first introduced in [1] as the first version of SimVP. It employs the inception module (see Fig. 3a) that consists of a bottleneck convolutional layer with 1×1 kernel, followed by parallel convolutional operators with kernels of increasing size. This combination allows the module to learn both local and global features. The whole spatiotemporal translator is made of $N_t$ Inception modules organized in an Unet-like architecture, as shown in Fig. 3b. The input latent representation is first processed by several Inception temporal modules from top to bottom and then by a symmetric path from bottom to top. The corresponding modules along the two paths have concatenation connections.

## 3.2 MetaFormer

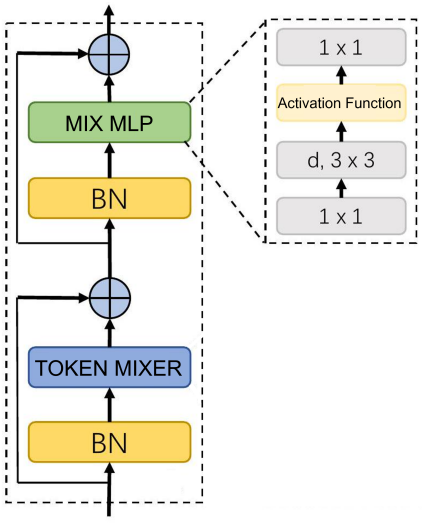
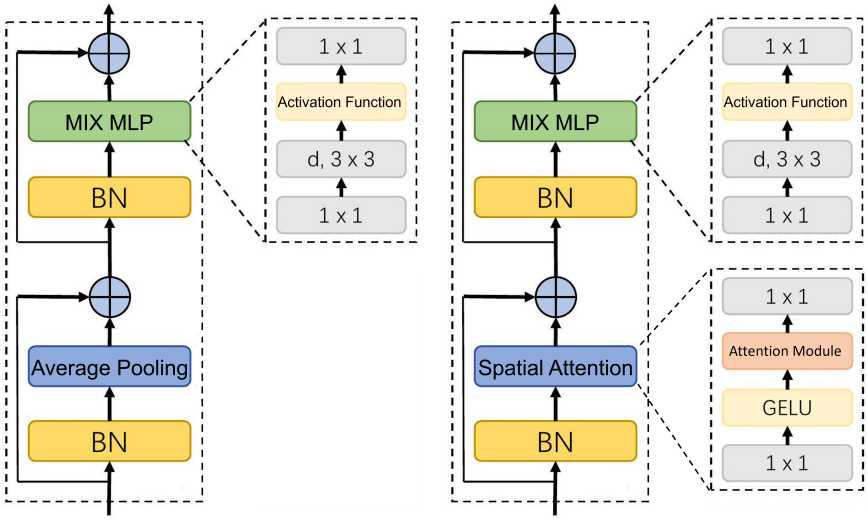



Figure 4: The mix MLP module is explicitly unfolded and contains: $1 \times 1$ convolutional layer, $3 \times 3$ depth-wise convolutional layer followed by an activation function that will be later specified, and finally another $1 \times 1$ convolutional layer.

The general MetaFormer abstraction (see Fig. 4) is introduced in [11], and it is empirically found that the success of Transformers actually lies in such structure, and not in the attention modules employed. In particular, as shown in Fig. 4, the overall architecture consists of two main sub-blocks: the first one contains a token mixer for mixing information among input tokens, and the second one contains a MLP block that implements a feedforward convolutional network [10]. In both blocks the main component is preceded by a Normalization layer and a residual connections is established around each.

### 3.2.1 PoolFormer




(a) PoolFormer (b) SimVPv2 MetaFormer

Figure 5: Two MetaFormer instantiations that are used in this work. In (a) a simple non parametric pooling operation is used, while in (b) an attention-based token mixer shall capture spatiotemporal dependencies.

An extremely simple instantiation of the MetaFormer is called PoolFormer, where the token mixer is a non learnable 2D average Pooling operation (see Fig. 5a). Thus, the second examined architecture of this work has as temporal translator a sequence of $N_t$ PoolFormer modules.

### 3.2.2 Gated SpatioTemporal Attention

The attention mechanism, widely exploited in visual transformers, allows to extract spatiotemporal features from the latent space representation. Large kernel convolutions share advantages with the attention mechanism, as they both result in large receptive fields: for this reason, to imitate the attention mechanism, large kernel convolutions can be employed instead. However, due to the large computational overhead, a $K \times K$ convolution is reduced in a sequence of smaller convolutions as follows

- $(2d-1) \times (2d-1)$ depth wise convolution that captures local receptive fields within a single channel
- $\frac{K}{d} \times \frac{K}{d}$ dilation depth wise convolution with dilation $d$ that builds connections between distant receptive fields
- $1 \times 1$ convolution to have interaction between channels.

This decomposition drastically reduces the number of parameters, keeping the same receptive field [2].

The last $1 \times 1$ convolution doubles the number of channels of the tensor, that is then split along the channel dimension in two chunks: one undergoes a sigmoid operation and acts as a gate, that is then multiplied element-wise with the other chunk. This gated spatiotemporal attention block is the Attention Module in the MetaFormer of Fig. 5b. A stack of $N_t$ of these MetaFormers composes the translator of the third examined architecture, called SimVPv2, introduced in [8].

# 4 Loss function

The loss function is composed of a main term that is the Mean Squared Error $\mathcal{L}_{MSE}$ between the predicted and true future frames. Additionally, to account for more than pixel-wise errors, two terms are added:

- *Gradient Difference Loss (GDL).* This criterion is first introduced in [4] as simply the error between the absolute value of the image gradients in the predicted and true frames, focusing on intra-frame variations. The GDL between two frames $I, \hat{I}$ can be written as

$$GDL(I, \hat{I}) = \frac{1}{CHW} \sum_{i}^{C} \sum_{j}^{H} \sum_{k}^{W} (|I_x|_{i,j,k} - |\hat{I}_x|_{i,j,k})^2 + (|I_y|_{i,j,k} - |\hat{I}_y|_{i,j,k})^2 \tag{1}$$

  where $I_x, I_y$ are the gradients of image $I$ along the $x$ and $y$ direction respectively. In this work, the gradients are computed using a Sobel operator through the Kornia library [5]. The loss term $\mathcal{L}_{GDL}$ is an average of such criterion along all the future frames in the current batch.

- *Differential Divergence Regularization (DDR).* This regularization term is introduced in [9] to learn the differences between consecutive frames, focusing on inter-frame variations. Given the true and predicted future frames $\mathcal{Y}, \hat{\mathcal{Y}} \in \mathbb{R}^{T' \times C \times H \times W}$, firstly compute the forward differences $\Delta\mathcal{Y} = \mathcal{Y}_{i+1} - \mathcal{Y}_i \in \mathbb{R}^{T'-1 \times C \times H \times W}$, and similarly $\Delta\hat{\mathcal{Y}}$. Then, these differences are transformed into probability distributions $\sigma(\Delta\mathcal{Y}), \sigma(\Delta\hat{\mathcal{Y}})$ using the softmax function on $C, H, W$ dimensions. Finally, the difference between such distribution is computed via the Kullback-Leibler divergence as

$$D_{KL}(\sigma(\Delta\hat{\mathcal{Y}}) || \sigma(\Delta\mathcal{Y})) = \sum_{i=1}^{T'-1} \sigma(\Delta\hat{\mathcal{Y}}_i) log \frac{\sigma(\Delta\hat{\mathcal{Y}}_i)}{\sigma(\Delta\mathcal{Y}_i)} \tag{2}$$

  and the loss regularization term $\mathcal{L}_{DDR}$ is the mean of this term over the batch dimension.

Finally, the overall loss function is

$$\mathcal{L} = \mathcal{L}_{MSE} + \lambda_1 \mathcal{L}_{GDL} + \lambda_2 \mathcal{L}_{DDR} \tag{3}$$

with weights $\lambda_1, \lambda_2 \geq 0$.

# 5 Experiments

All the experiments are performed on the Moving MNIST dataset, which is a dataset made of sequences of 20 frames where two digits from the MNIST dataset move inside a $64 \times 64$ grayscale image. The sequences are split having 10 frames in input, and 10 frames as output to predict. The training settings of the following experiments, unless otherwise specified, are the following:

- as in [8], all the models are trained using the Adam optimizer and with the OneCycle learning rate scheduler, starting from a learning rate of 0.001
- the models are trained for 100 epochs using a batch size of 32
- the structural parameters $N_t, N_s$ are set to 4
- the number of channels in the encoder and decoder $C_s$ is set to 32, the number of channels in the translator blocks $C_t$ is set to 256.

Evaluation is based on the MSE — the lower the better — and the mean Structural Similarity Index (SSIM) of the future frames — a number in $[0, 1]$, the higher the better — to compare the prediction and the ground truth frames beyond pixel-wise differences.

In the follwing experiments, various setups are firstly analyzed for SimVPv2, and then a comparison with the two baselines, the PoolFormer and Inception Unet architectures, is carried out.

## 5.1 SimVPv2

In this section, SimVPv2 is investigated more in depth by analyzing the following:

- the influence of data augmentation during training
- the influence of two different activation functions, GELU and StarReLU, in the MetaFormer Feed Forward block (see Fig. 5b)
- the influence of $\mathcal{L}_{GDL}, \mathcal{L}_{DDR}$ in the loss function.

**Data Augmentation** The analyzed data augmentation techniques are random cropping and horizontal/vertical flipping. From Table 1 it is clear that these kind of augmentation are not well suited for this task on this dataset, hence all next experiments will be done without any type of data augmentation. The experiments of Table 1 are all done using the GELU activation function, and with $\lambda_1 = \lambda_2 = 0$ in Eq. (3).

| Data Augmentation | MSE ↓ | SSIM ↑ |
|---|---|---|
| Crop and Flip | 72.3073 | 0.8364 |
| Crop only | 71.7384 | 0.83259 |
| Disabled | **41.2836** | **0.90222** |

Table 1: Comparison between Data Augmentation techniques. The model is SimVPv2 trained for 100 epochs, with a batch size of 32, using the GELU activation function and the simple MSE as loss function.

**Activation Function** Two different activation functions are used and compared, namely GELU and StarReLU. The GELU [3] (Gaussian Error Linear Units) function is defined as $GELU(x) = x\Phi(x)$ where $\Phi(x)$ is the Cumulative Distribution Function for Gaussian Distributions. On the other hand, StarReLU [12] is defined as $StarReLU(x) = s \cdot (ReLU(x))^2 + b$ where $s, b \in \mathbb{R}$ are shared between all the channels and can be either learnable or fixed — in this work they are always learnable. This function introduces less computational overhead with respect to GELU and, as can be seen in Table 2, it leads to better results. Hence, all next experiments will use the StarReLU activation function. As in the previous experiments, also here the loss function in Eq. (3) contains only the MSE term.

| Data Augmentation | MSE ↓ | SSIM ↑ |
|---|---|---|
| GELU | 41.2836 | 0.90222 |
| StarReLU | **41.2322** | **0.90244** |

Table 2: Comparison between the two different activation functions. The model is SimVPv2 trained for 100 epochs, with a batch size of 32 and the simple MSE as loss function.

| Additional Loss term | $\{\lambda_1,\ \lambda_2\}$ | MSE ↓ | SSIM ↑ |
|---|---|---|---|
| None | {0, 0} | 41.2322 | 0.90244 |
| $\mathcal{L}_{GDL}$ | {0.5, 0} | 41.0523 | **0.90474** |
| $\mathcal{L}_{GDL}$, $\mathcal{L}_{DDR}$ | {0.5, 0.1} | **40.9828** | 0.90452 |

Table 3: Comparison between different loss functions. The model is SimVPv2 trained for 100 epochs, with a batch size of 32.

**Loss function** Additional trials are done to investigate whether the presence of the intra-frame loss term, $\mathcal{L}_{GDL}$, and inter-frame regularization term, $\mathcal{L}_{DDR}$, in the loss function can help to achieve better performances. The results are reported in Table 3, and it emerges that $\mathcal{L}_{GDL}$ brings a performance improvement, so all the following experiments will be done with $\lambda_1 = 0.5$. However, it is not clear whether $\mathcal{L}_{DDR}$ is useful. To explore this further, additional experiments were carried out and the results are reported in Table 4. In these experiments, the number of epochs, batch size and $\lambda_2$ are varied to identify the best setup. From the different sub-blocks of Table 4, it is possible to draw the following conclusion: the term $\mathcal{L}_{DDR}$ does not have a hard-hitting and clear contribution; a smaller batch size of 16 improves the performance in every case; finally, as expected, doubling the number of epochs results in significantly better performances.

| Epochs | Batch size | $\lambda_2$ | MSE ↓ | SSIM ↑ |
|---|---|---|---|---|
| 100 | 32 | 0 | 41.0523 | <u>0.90474</u> |
| 100 | 32 | 0.1 | <u>40.9828</u> | 0.90452 |
| 100 | 32 | 0.5 | 41.0197 | 0.90448 |
| 100 | 16 | 0 | 40.0263 | <u>0.90715</u> |
| 100 | 16 | 0.1 | <u>40.01</u> | 0.90706 |
| 200 | 16 | 0 | **34.4117** | 0.92243 |
| 200 | 16 | 0.1 | 34.4377 | **0.92262** |

Table 4: Results of different experiments aimed at understanding the influence of the $\mathcal{L}_{DDR}$ term in the loss function considering the SimVPv2 model. Comparisons are made varying the value of the $\lambda_2$ hyperparameter in Eq. (3), the number of training epochs and the batch size. The underlined results in each block of the table identify the best performance in that block.

## 5.2 Models Comparison

In this Section the performance of the SimVPv2 model is compared against the two baselines, PoolFormer and Inception Unet, considering the best setups from Table 4, i.e., 200 training epochs, batch size of 16 and $\lambda_2 \in \{0, 0.1\}$. The comparison is reported in Table 5. As expected, the extremely simple PoolFormer has the worst overall results and the least number of parameters. On the other hand, despite having a higher number of parameters, the Inception Unet model is noticeably outpeformed by SimVPv2. Note that for all the models, the term $\mathcal{L}_{DDR}$ does not have a clear influence. In Table 5, another model is introduced: namely, SimVPv2-L refers to a larger SimVPv2 structure where the parameters $N_t, C_s, C_t$ are all doubled with respect to the settings used in all the other experiments.

In Table 6 an example of the prediction is reported for the various models from Table 5. In particular, for every model, the setup resulting in the lowest MSE is considered.

| Model | Parameters ≈ (millions) | $\lambda_2$ | MSE ↓ | SSIM ↑ |
|---|---|---|---|---|
| PoolFormer | 5.2 | 0 | 42.5548 | <u>0.90001</u> |
| PoolFormer | 5.2 | 0.1 | <u>42.527</u> | 0.8997 |
| Inception Unet | 7.4 | 0 | <u>39.1267</u> | <u>0.9119</u> |
| Inception Unet | 7.4 | 0.1 | 39.1313 | 0.9118 |
| SimVPv2 | 6.5 | 0 | <u>34.4117</u> | 0.92243 |
| SimVPv2 | 6.5 | 0.1 | 34.4377 | <u>0.92262</u> |
| SimVPv2-L | 46.8 | 0 | **29.8718** | **0.93976** |

Table 5: Comparison between the three different models varying the value of $\lambda_2$ hyperparameter of Eq. (3). The model SimVPv2-L is a model with the SimVPv2 architecture, but with a different architecture depth and width with respect to the other reported models. The number of training epochs and batch size are, respectively, 200 and 16. The underlined results in each block of the table identify the best performance in that block.

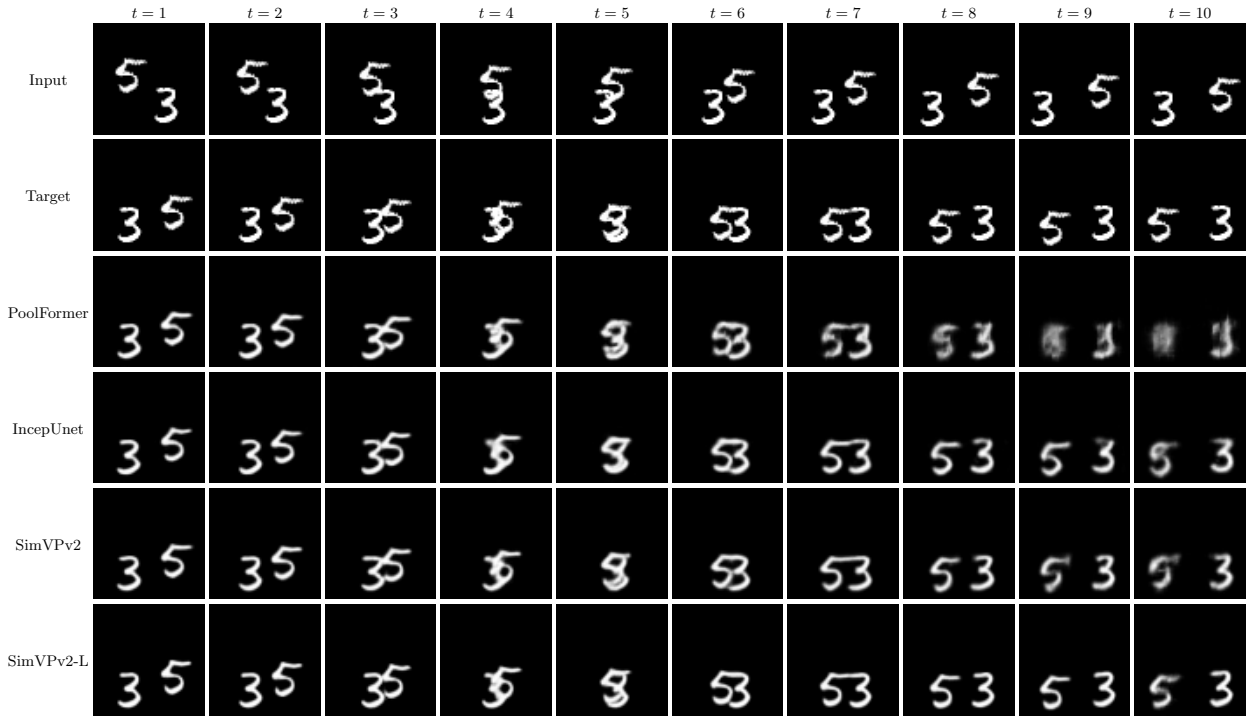



Table 6: Examples of predicted results on the Moving MNIST dataset. We denote the Inception Unet model as IncepUnet for convenience here. Input denotes the sequence of 10 input frames, while Target denotes the ground truth future frames.

# 6 Conclusions

In this work, a fully convolutional architecture, called SimVP, is presented for video prediction task. SimVP challenges the common sense that pure convolution networks are weak in capturing spatiotemporal correlations: indeed, as reported in [8], when trained for enough epochs it achieves SOTA performances. Moreover, it shows high flexibility as it can accomodate any MetaFormer instantiation, as shown with the PoolFormer model.

The performed experiments show that the analyzed data augmentation techniques (see Table 1) are not

suitable for this dataset, possibly due to the fact that a cropped and/or flipped digit is never encountered during testing, hence applying such transformations during training results in useless generalization capabilities of the network.

Furthermore, it has been shown that the Gradient Difference Loss (GDL) term brings better performances with no additional overhead, differently from the Differential Divergence Regularization (DDR) which brings contrasting results.

Finally, as shown in Table 6, it is possible to obtain sharper predictions with a model of higher capacity, at the cost of raising the training time.

# References


[1] Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z. Li. Simvp: Simpler yet better video prediction, 2022.

[2] Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network, 2022.

[3] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[4] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error, 2016.

[5] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Winter Conference on Applications of Computer Vision*, 2020.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[8] Cheng Tan, Zhangyang Gao, Siyuan Li, and Stan Z. Li. Simvp: Towards simple yet powerful spatiotemporal predictive learning, 2023.

[9] Cheng Tan, Zhangyang Gao, Lirong Wu, Yongjie Xu, Jun Xia, Siyuan Li, and Stan Z. Li. Temporal attention unit: Towards efficient spatiotemporal predictive learning, 2023.

[10] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. PVT v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):415–424, mar 2022.

[11] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision, 2022.

[12] Weihao Yu, Chenyang Si, Pan Zhou, Mi Luo, Yichen Zhou, Jiashi Feng, Shuicheng Yan, and Xinchao Wang. Metaformer baselines for vision, 2022.