

Folded Recurrent Neural Networks for Future Video Prediction

Marc Oliu^{1,3}, Javier Selva^{2,3}, and Sergio Escalera^{2,3}

¹ Universitat Oberta de Catalunya

Rambla del Poblenou, 156, 08018 Barcelona, Spain
moliusion@uoc.edu

² Universitat de Barcelona

Gran Via de les Corts Catalanes, 585, 08007 Barcelona
jselvaca21@alumnes UB.edu, sescalera@ub.edu

³ Centre de Visió per Computador

Campus UAB, Edifici O, 08193 Cerdanyola del Vallès, Barcelona

Abstract. This work introduces double-mapping Gated Recurrent Units (dGRU), an extension of standard GRUs where the input is considered as a recurrent state. An extra set of logic gates is added to update the input given the output. Stacking multiple such layers results in a recurrent auto-encoder: the operators updating the outputs comprise the encoder, while the ones updating the inputs form the decoder. Since the states are shared between corresponding encoder and decoder layers, the representation is stratified during learning: some information is not passed to the next layers. We test our model on future video prediction. Main challenges for this task include high variability in videos, temporal propagation of errors, and non-specificity of future frames. We show how only the encoder or decoder needs to be applied for encoding or prediction. This reduces the computational cost and avoids re-encoding predictions when generating multiple frames, mitigating error propagation. Furthermore, it is possible to remove layers from a trained model, giving an insight to the role of each layer. Our approach improves state of the art results on MMNIST and UCF101, being competitive on KTH with 2 and 3 times less memory usage and computational cost than the best scored approach.

Keywords: Future video prediction · unsupervised learning · recurrent neural networks

1 Introduction

Future video prediction is a challenging task that recently received much attention due to its capabilities for learning in an unsupervised manner, making it possible to leverage large volumes of unlabelled data for video-related tasks such as action and gesture recognition [22, 11, 10], task planning [14, 4], weather prediction [20], optical flow estimation [15] and new view synthesis [10].

One of the main problems in this task is the need of expensive models, both in terms of memory and computational power, in order to capture the variability present in video data. Another problem is the propagation of errors in recurrent models, which is tied to the inherent uncertainty of video prediction: given a series of previous frames, there are multiple feasible futures. Left unchecked, this results in blurry predictions averaging the space of possible futures. When predicting subsequent frames, the blur is propagated back into the network, accumulating errors over time.

In this work we propose a new type of recurrent auto-encoder (AE) with state sharing between encoder and decoder. We show how the exposed state in Gated Recurrent Units (GRU) can be used to create a bidirectional mapping between the input and output of each layer. To do so, the input is treated as a recurrent state, adding another set of logic gates to update it based on the output. Creating a stack of these layers allows for a bidirectional flow of information: The forward gates encode inputs and the backward ones generate predictions, obtaining a structure similar to an AE⁴, but with many inherent advantages. Only the encoder or decoder is executed for input encoding or prediction, reducing memory and computational costs. Furthermore, the representation is stratified: low level information not necessary to capture higher level dynamics is not passed to the next layer. Also, it naturally provides a noisy identity mapping of the input, facilitating the initial stages of training. While the approach does not solve the problem of blur, it prevents its magnification by mitigating the propagation of errors. Moreover, a trained network can be deconstructed to analyse the role of each layer in the final predictions, making the model more explainable. Since the states are shared, the architecture can be thought of as a recurrent AE folded in half, with encoder and decoder layers overlapping. We call our method Folded Recurrent Neural Network (fRNN).

Our main contributions are: 1) A new shared-state recurrent AE with lower memory and computational costs. 2) Mitigation of error propagation through time. 3) It naturally provides an identity function during training. 4) Model explainability and optimisation through layer removal. 5) Demonstration of representation stratification.

2 Related work

Video prediction is usually approached using deep recurrent models. While initial proposals focused on predicting small patches [17, 13], it is now common to generate the whole frame based on the previous ones.

Building Blocks. Due to the characteristics of the problem, an AE setting has been widely used [22, 14, 5, 24, 3]: the encoder extracts information from the input and the decoder produces new frames. Usually, encoder and decoder are CNNs that tackle the spatial dimension. LSTMs are commonly used to handle the temporal dynamics and project the representations into the future. Some

⁴ Code available at <https://github.com/moliusimon/frnn>

works compute the temporal dynamics at the deep representation bridging the encoder and decoder [14, 15, 2, 3]. Others jointly handle space and time by using Convolutional LSTMs [5, 11, 15, 8, 9] (or GRUs, as in our case), which use convolutional kernels at their gates. For instance, Lotter *et al.* [11] use a recurrent residual network with ConvLSTMs where each layer minimises the discrepancies from previous block predictions. Common variations also include a conditional term to guide the temporal transform, such as a time differential [25] or prior knowledge of scene events, reducing the space of possible futures. Oh *et al.* [14] predict future frames on Atari games conditioning on the player action. Some works propose such action conditioned models foreseeing an application for autonomous agents learning in an unsupervised fashion [5, 8]. Finn *et al.* [5] condition their predictions for a physical system on the actions taken by a robotic arm interacting with the scene. The method was recently applied to task planning [4] and adapted to stochastic future video prediction [1].

Bridge connections. Introducing bridge connections (connections between equivalent layers of the encoder and decoder) is also common [5, 10, 2, 24]. This allows for a stratified representation of the input sequence, reducing the capacity needs of subsequent layers. *Video Ladder Networks* (VLN) [2] use a conv. AE where pairs of convolutions are grouped into residual blocks. Bridge connections are added between corresponding blocks, both directly and by using a recurrent bridge layer. This topology was further extended with *Recurrent Ladder Networks* (RLN) [16], where the recurrent bridge connections were removed, and the residual blocks replaced by recurrent layers. We propose an alternative to bridge connections by completely sharing the state between encoder and decoder.

Prediction atom. Most of the proposed architectures for future frame prediction work at the pixel level. However, some models have been designed to predict motion and use it to project the last frame into the future. These may generate optical flow maps [10, 15] or conv. kernels [7, 27]. Other methods propose mapping the input sequence onto predefined feature spaces, such as affine transforms [23] or human pose vectors [26]. These systems use sequences of such features to generate the next frame at the pixel level.

Loss and GANs. Commonly used loss functions such as L2 or MSE tend to average the space of possible futures. For this reason, some works [12, 24, 26, 9] propose using Generative Adversarial Networks (GAN) [6] to aid in the generation of realistic looking frames and coherent sequences. Mathieu *et al.* [12] use a plain multi-scale CNN in an adversarial setting and propose the Gradient Difference Loss to sharpen the predictions.

Disentangled Motion/Content. Some authors encode content and motion separately. Villegas *et al.* [24] use an AE architecture with a two-stream encoder: for motion, a CNN + LSTM encodes difference images; for appearance, a CNN encodes the last input frame. In a similar fashion, Denton *et al.* [3] use two separate encoders and an adversarial setting to obtain a disentangled representation of content and motion. Alternatively, some works predict motion and content in parallel to benefit from the combined strengths of both tasks. While Sedaghat *et al.* [19] propose using a single representation with a dual objective (optical

flow and future frame prediction), Liang *et al.* [9] use a dual GAN setting and use predicted motion to refine the future frame prediction.

Feedback Predictions. Finally, most recurrent models are based on the use of feedback predictions: they input previous predictions to generate subsequent frames. If not handled properly, this may cause errors to accumulate and magnify over time. Our model mitigates this by enabling encoder and decoder to be executed any number of times independently. This is similar to the proposal by Srivastava *et al.* [22], which uses a recurrent AE approach where an input sequence is encoded and its state copied into the decoder. The decoder is then applied to generate a given number of frames. However, it is limited to a single recurrent layer at each part.

Here, stochastic video prediction is not considered. Such models learn and sample from a space of possible futures to generate the following frames. This reduces prediction blur by preventing the averaging of possible futures. fRNN could be extended to perform stochastic sampling by adding an inference model similar to that in [1] during training. Samples drawn from the predicted distribution would be placed into the deepest state of the dGRU stack. However, this would make it difficult to analyse the contribution of dGRU layers to the mitigation and recovery from blur propagation.

3 Proposed method

We propose an architecture based on recurrent conv. AEs to deal with the network capacity and error propagation problems for future video prediction. It is built by stacking multiple double-mapping GRU layers, which allow for a bidirectional flow of information between input and output: they consider the input as a recurrent state and update it using an extra set of gates. These are then stacked, forming an encoder and decoder using, respectively, the forward and backward gates (Fig.1). We call this architecture Folded Recurrent Neural Network (fRNN). Because of the state sharing between encoder and decoder, the topology allows for: stratification of the representation, lower memory and computational requirements compared to regular recurrent AEs, mitigated propagation of errors, and increased explainability through layer removal.

3.1 Double-mapping Gated Recurrent Units

GRUs have their state fully exposed as output. This allows us to define a bidirectional mapping between input and output by replicating the logic gates of the GRU layer. To do so, we consider the input as a state. Lets define the output of a GRU at layer l and time step t as $h_t^l = f_f^l(h_t^{l-1}, h_{t-1}^l)$ given an input h_t^{l-1} and its state at the previous time step h_{t-1}^l . A second set of weights can be used to define an inverse mapping $h_t^{l-1} = f_b^l(h_t^l, h_{t-1}^{l-1})$ using the output of the forward function at the current time step to update its input, which is treated as the hidden state of the inverse function. This is illustrated in Fig. 1. We will refer to this bidirectional mapping as a double-mapping GRU (dGRU).

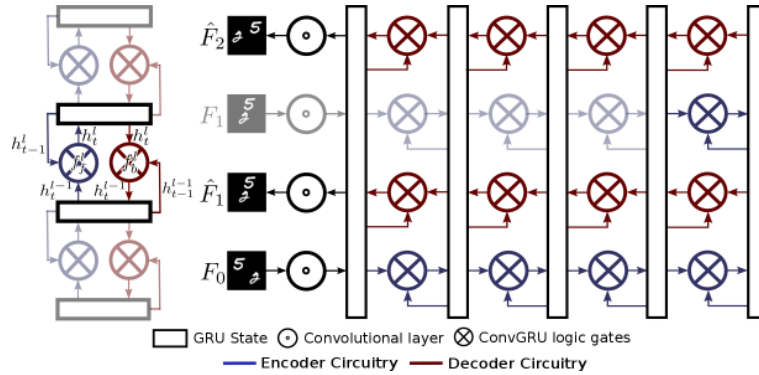


Fig. 1. Left: Scheme of a dGRU. Shaded areas illustrate additional dGRU layers. **Right:** fRNN topology. State cells are shared between encoder and decoder, creating a bidirectional state mapping. Shaded areas represent unnecessary circuitry: re-encoding of the predictions is avoided due to the decoder updating all the states.

3.2 Folded Recurrent Neural Network

By stacking multiple dGRUs, a recurrent AE is obtained. Given n dGRUs, the encoder is defined by the set of forward functions $E = \{f_f^1, \dots, f_f^n\}$ and the decoder by the set of backward functions $D = \{f_b^n, \dots, f_b^1\}$. This is illustrated in Fig. 1, and is equivalent to a recurrent AE, but with shared states, having 3 main advantages: 1) It is not necessary to feed the predictions back into the network in order to generate the following predictions. Because of state sharing, the decoder already updates all the states except for the bridge state between encoder and decoder, which is updated by applying the last layer of the encoder before decoding. The shaded area in Fig. 1 shows the section of the computational graph that is not required when performing multiple sequential predictions. For the same reason, when considering multiple sequential elements before prediction, only the encoder is required. 2) Since the network updates its states from the higher level representations to the lowest ones during prediction, errors introduced at a given layer are not propagated into deeper layers, leaving higher-level dynamics unaffected. 3) The model implicitly provides a noisy identity function during training: the input state of the first dGRU layer is either the input image itself, when preceded by conv. layers, or an over-complete representation of the same. A noise signal is then introduced to the representation by the backward function of the untrained first dGRU layer. This is exemplified in Fig. 7, when all dGRU layers are removed. As shown in Section 4.3, this helps the model to converge on MMNIST: when the same background is shared across instances, it prevents the model from killing the gradients by adjusting the biases to match the background and setting the weights to zero.

This approach shares some similarities with VLN [2] and RLN [16]. As with them, part of the information can be passed directly between corresponding layers of the encoder and decoder, not having to encode a full representation

of the input into the deepest layer. However, our model implicitly passes the information through the shared recurrent states, making bridge connections unnecessary. When compared against an equivalent recurrent AE with bridge connections, this results in lower computational and memory costs. More specifically, the number of weights in a pair of forward and backward functions is equal to $3(\overline{h^{l-1}}^2 + \overline{h^l}^2 + 2\overline{h^{l-1}} \overline{h^l})$ in the case of dGRU, where $\overline{h^l}$ corresponds to the state size of layer l . When using bridge connections, that value is increased to $3(\overline{h^{l-1}}^2 + \overline{h^l}^2 + 4\overline{h^{l-1}} \overline{h^l})$. This corresponds to an overhead of 44% in the number of parameters when one state has double the size of the other, and of 50% when they have the same size. Furthermore, both the encoder and decoder must be applied at each time step. Thus, memory usage is doubled and computational cost is increased by a factor of between 2.88 and 3.

3.3 Training Folded RNNs

We propose a training approach for fRNNs that exploits their ability to skip the encoder or decoder at a given time step. First g ground truth frames are passed to the encoder. The decoder is then applied p times, producing p predictions. This uses up only half the memory: either encoder or decoder is applied at each step, never both. This has the same advantage as the approach by Srivastava [22], where recurrently applying the decoder without further ground truth inputs encourages the network to learn video dynamics. This also prevents the network from learning an identity model, i.e. copying the last input to the output.

4 Experiments

In this section, we first discuss the data, evaluation protocol, and methods. We then provide quantitative and qualitative evaluations. We finish with a brief analysis on the stratification of the sequence representation among dGRU layers.

4.1 Data and evaluation protocol

Three datasets of different complexity are considered: Moving MNIST (MMNIST)[22], KTH [18], and UCF101 [21]. MMNIST consists of 64×64 grayscale sequences of length 20 displaying pairs of digits moving around the image. We generated a million training samples by randomly sampling pairs of digits and trajectories. The test set is fixed and contains 10000 sequences. KTH consists of 600 videos of 15-20 seconds with 25 subjects performing 6 actions in 4 different settings. The videos are grayscale, at a resolution of 120×160 pixels and 25 fps. The dataset has been split into subjects 1 to 16 for training, and 17 to 25 for testing, resulting in 383 and 216 sequences, respectively. Frame size is reduced to 64×80 by removing 5 pixels from the left and right borders and using bilinear interpolation. UCF101 displays 101 actions, such as playing instruments, weight lifting or sports. It is the most challenging dataset considered, with a high intra-class variability. It contains 9950 training and 3361 test sequences. These are

	Conv 1	Conv 2	Pool 1	dGRU 1	dGRU 2	Pool 2	dGRU 3	dGRU 4	Pool 3	dGRU 5	dGRU 6	Pool 4	dGRU 7	dGRU 8
Num. Units	32	64	-	128	128	-	256	256	-	512	512	-	256	256
Kernel size	5×5	5×5	2×2	5×5	5×5	2×2	5×5	5×5	2×2	3×3	3×3	2×2	3×3	3×3
Stride	1	1	2	1	1	2	1	1	2	1	1	2	1	1
Activation	tanh	tanh	-	sigmoid & tanh	-	-	sigmoid & tanh	-	-	sigmoid & tanh	-	-	sigmoid & tanh	-

Table 1. Parameters of the topology used for the experiments. The decoder applies the same topology in reverse, using nearest neighbours interpolation and transposed convolutions to revert the pooling and convolutional layers.

RGB at a resolution of 320×240 pixels and 25 fps. The frame size is reduced to 64×85 and the frame rate halved to magnify frame differences.

All methods are tested using 10 input frames to generate the following 10 frames. We use 3 common metrics for video prediction analysis: Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Dissimilarity (DSSIM). MSE and PSNR are objective measurements of reconstruction quality. DSSIM is a measure of the perceived quality. For DSSIM we use a Gaussian sliding window of size 11×11 and $\sigma = 1.5$.

4.2 Methods

The proposed method was trained using RMSProp with a learning rate of 0.0001 and a batch size of 12, sampling a random sub-sequence at each epoch. Weights were orthogonally initialised and biases set to 0. For testing, all sub-sequences of length 20 were considered. Our network topology consists of two convolutional layers followed by 8 convolutional dGRU layers, applying a 2×2 max pooling every 2 layers. Topology details are shown in Table 1. The convolutional and max pooling layers are reversed by using transposed convolutions and nearest neighbours interpolation, respectively. We train with an L1 loss.

For evaluation, we include a stub baseline model predicting the last input frame, and a second baseline (RLadder) to evaluate the advantages of using state sharing. RLadder has the same topology as the fRNN model, but uses bridge connections instead of state sharing. Note that to keep the same state size on ConvGRU layers, using bridge connections doubles the memory size and almost triples the computational cost (Sec.3.2). This is similar to how RLN [16] works, but using regular ConvGRU layers in the decoder. We also compare against Srivastava [22] and Mathieu [12]. The former handles only the temporal dimension with LSTMs, while the latter uses a 3D-CNN, not providing memory management mechanisms. Next, we compare against Villegas [24], which, contrary to our proposal, uses feedback predictions. Finally, we compare against Lotter *et al.* [11] which is based on residual error reduction. All of them were adapted to train using 10 frames as input and predicting the next 10, using the topologies and parameters defined by the authors.

4.3 Quantitative analysis

The first row of Fig. 2 displays the results for the MMNIST dataset for the considered methods. Mean scores are shown in Table 2. fRNN performs best on

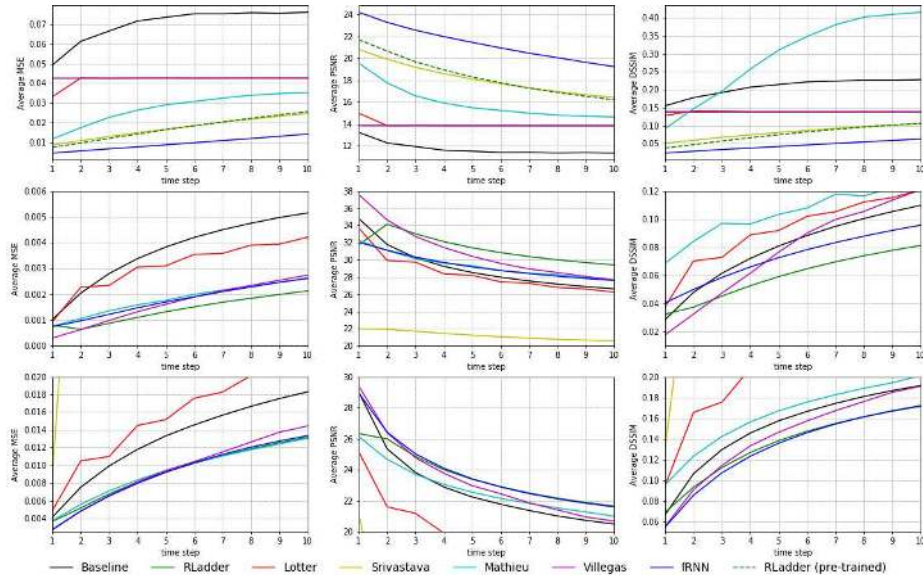


Fig. 2. Quantitative results on the considered datasets in terms of the number of time steps since the last input frame. From top to bottom: MMNIST, KTH, and UCF101. From left to right: MSE, PSNR, and DSSIM. For MMNIST, RLadder is pre-trained to learn an initial identity mapping, allowing it to converge.

all time steps and metrics, followed by Srivastava *et al.* [22]. These two are the only methods to provide valid predictions on this dataset: Mathieu *et al.* [12] progressively blurs the digits, while the other methods predict a black frame. This is caused by a loss of gradient during the first training stages. On more complex datasets the methods start by learning an identity function, then refining the results. This is possible since in many sequences most of the frame remains unchanged. In the case of MMNIST, where the background is homogeneous, it is easier for the models to set the weights of the output layer to zero and set the biases to match the background colour. This truncates the gradient and prevents further learning. Srivastava *et al.* [22] use an auxiliary decoder to reconstruct the input frames, forcing the model to learn an identity function. This, as discussed at the end of Section 3.2, is implicitly handled in our method, giving an initial solution to improve on and preventing the models from learning a black image. In order to verify this effect, we pre-trained RLadder on the KTH dataset and then fine-tuned it on the MMNIST dataset. While KTH has different dynamics, the initial step to solve the problem remains: providing an identity function. As shown in Fig. 2 (dashed lines), this results in the model converging, with an accuracy comparable to Srivastava *et al.* [22] for the 3 evaluation metrics.

On the KTH dataset, Table 2 shows the best approach is our RLadder baseline followed by fRNN and Villegas *et al.* [24], both having similar results, but with Villegas *et al.* having slightly lower MSE and higher PSNR, and fRNN a

	MMNIST			KTH			UCF101		
	MSE	PSNR	DSSIM	MSE	PSNR	DSSIM	MSE	PSNR	DSSIM
Baseline	0.06989	11.745	0.20718	0.00366	29.071	0.07900	0.01294	22.859	0.15043
RLadder	0.04254	13.857	0.13788	0.00139	31.268	0.05945	0.00918	23.558	0.13395
Lotter [11]	0.04161	13.968	0.13825	0.00309	28.424	0.09170	0.01550	19.869	0.21389
Srivastava [22]	0.01737	18.183	0.08164	0.00995	21.220	0.19860	0.14866	10.021	0.42555
Mathieu [12]	0.02748	15.969	0.29565	0.00180	29.341	0.10410	0.00926	22.781	0.16262
Villegas [24]	0.04254	13.857	0.13896	0.00165	30.946	0.07657	0.00940	23.457	0.14150
fRNN	0.00947	21.386	0.04376	0.00175	29.299	0.07251	0.00908	23.872	0.13055

Table 2. Average results over 10 time steps.

lower DSSIM. While both approaches obtain comparable average results, the error increases faster over time in the case of Villegas *et al.* (second row in Fig.2). Mathieu obtains good scores for MSE and PSNR, but has a much worse DSSIM.

For the UCF101 dataset, Table 2, our fRNN approach is the best performing for all 3 metrics. At third row of Fig. 5 one can see that Villegas *et al.* start out with results similar to fRNN on the first frame, but as in the case of KTH and MMNIST, the predictions degrade faster. Two methods display low performance in most cases. Lotter *et al.* work well for the first predicted frame in the case of KTH and UCF101, but the error rapidly increases on the following predictions. This is due to a magnification of prediction artefacts, making the method unable to predict multiple frames without supervision. In the case of Srivastava *et al.* the problem is about capacity: it uses fully connected LSTM layers, making the number of parameters explode quickly with the state cell size. This severely limits the representation capacity for complex datasets such as KTH and UCF101.

Overall, for the considered methods, fRNN is the best performing on MMNIST and UCF101, the latter being the most complex of the 3 datasets. We achieved these results with a simple topology: apart from the proposed dGRU layers, we use conventional max pooling with an L1 loss. There are no normalisation or regularisation mechanisms, specialised activation functions, complex topologies or image transform operators. In the case of MMNIST, fRNN shows the ability to find a valid initial representation and converges to good predictions where most other methods fail. In the case of KTH, fRNN has an overall accuracy comparable to that of Villegas *et al.*, being more stable over time. It is only surpassed by the proposed RLadder baseline, a method equivalent to fRNN but with 2 and 3 times more memory and computational requirements.

4.4 Qualitative analysis

We evaluate our approach qualitatively on some samples from the three considered datasets. Fig. 3 shows the last 5 input frames from some MMNIST sequences along with the next 10 ground truth frames and their corresponding fRNN predictions. As shown, the digits maintain their sharpness across the sequence of predictions. Also, the bounces at the edges of the image are correctly predicted and the digits do not distort or deform when crossing each other. This shows

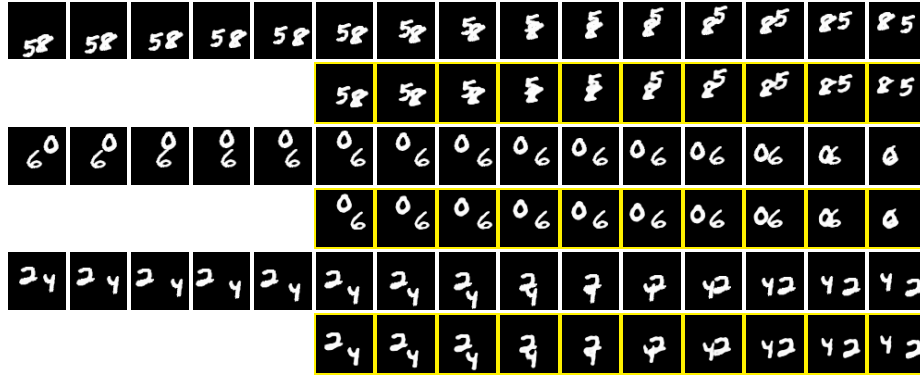


Fig. 3. fRNN predictions on MNIST. First row for each sequence shows last 5 inputs and target frames. Yellow frames are model predictions.

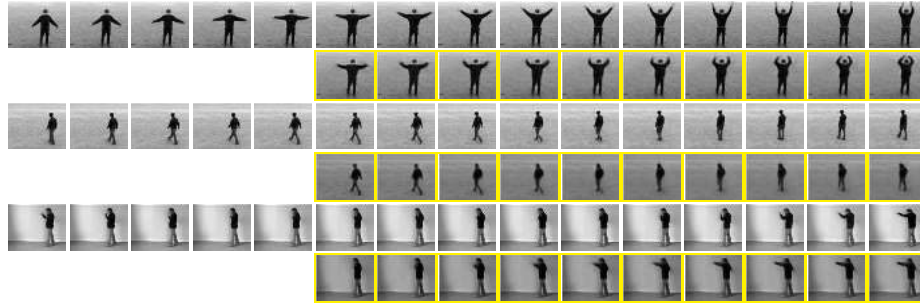


Fig. 4. fRNN predictions on KTH. First row for each sequence shows last 5 inputs and target frames. Yellow frames are model predictions.

the network internally encodes the appearance of each digit, facilitating their reconstruction after sharing the same region in the image plane.

Qualitative examples of fRNN predictions on the KTH dataset are shown in Fig. 4. It shows three actions: hand waving, walking, and boxing. The blur stops increasing after the first three predictions, generating plausible motions for the corresponding actions while background artefacts are not introduced. Although the movement patterns for each type of action have a wide range of variability on its trajectory, dGRU gives relatively sharp predictions for the limbs. The first and third examples also show the ability of the model to recover from blur. The blur slightly increases for the arms while the action is performed, but decreases again as these reach the final position.

Fig. 5 shows fRNN predictions on the UCF101 dataset. These correspond to two different physical exercises and a girl playing the piano. Common to all predictions, the static parts do not lose sharpness over time, and the background is properly reconstructed after an occlusion. The network correctly predicts actions with low variability, as shown in rows 1-2, where a repetitive movement is per-

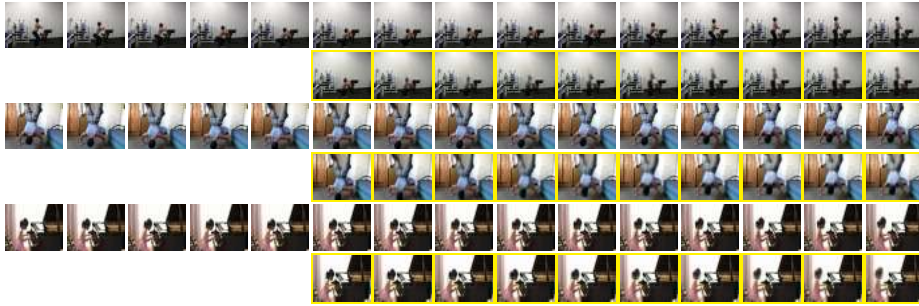


Fig. 5. fRNN predictions on UCF101. First row for each sequence shows last 5 inputs and target frames. Yellow frames are model predictions.

formed, and in the last row, where the girl recovers a correct body posture. Blur is introduced to these dynamic regions due to uncertainty, averaging the possible futures. The first row also shows an interesting behaviour: while the woman is standing up the upper body becomes blurry, but the frames sharpen again as the woman finishes her motion. Since the model does not propagate errors to deeper layers nor makes use of previous predictions for the following ones, the introduction of blur does not imply it will be propagated. In this example, while the middle motion could have multiple predictions depending on the movement pace and the inclination of the body, the final body pose has lower uncertainty.

In Fig. 6 we compare predictions from the proposed approach against the RLadder baseline and other state of the art methods. For the MMNIST dataset we do not consider Villegas *et al.* and Lotter *et al.* since these methods fail to successfully converge and they predict a sequence of black frames. From the rest of approaches, fRNN obtains the best predictions, with little blur or distortion. The RLadder baseline is the second best approach. It does not introduce blur, but heavily deforms the digits after they cross. Srivastava *et al.* and Mathieu *et al.* both accumulate blur over time, but while the former does so to a smaller degree, the latter makes the digits unrecognisable after five frames.

For KTH, Villegas *et al.* obtains outstanding qualitative results. It predicts plausible dynamics and maintains the sharpness of both the individual and background. Both fRNN and RLadder follow closely, predicting plausible dynamics, but not being as good at maintaining the sharpness of the individual. On UCF101, our model obtains the best predictions, with little blur or distortion compared to the other methods. The second best is Villegas *et al.*, successfully capturing the movement patterns but introducing more blur and important distortions on the last frame. When looking at the background, fRNN proposes a plausible initial estimate and progressively completes it as the woman moves. On the other hand, Villegas *et al.* modifies already generated regions as more background is uncovered, producing an unrealistic sequence. Srivastava *et al.* and Lotter *et al.* fail on both KTH and UCF101. Srivastava *et al.* heavily distort the frames. As discussed in Section 4.3, this is due to the use of fully connected

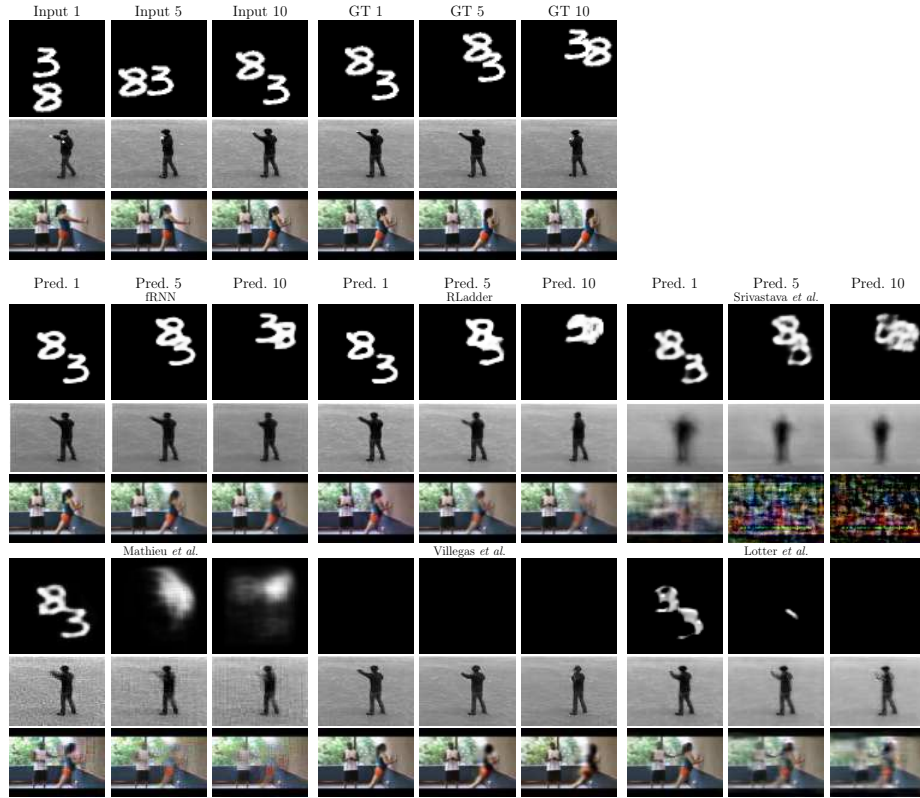


Fig. 6. Predictions at 1, 5, and 10 time steps from the last ground truth frame. RLadder predictions on MMNIST are from the model pre-trained on KTH.

recurrent layers, which constrains the state size and prevents the model from encoding relevant information on complex scenarios. In the case of Lotter *et al.*, it makes good predictions for the first frame, but rapidly accumulates artefacts.

4.5 Representation stratification analysis

Here we analyse the stratification of the sequence representation among dGRU layers. Because dGRU units allow for a bidirectional mapping between states, it is possible to remove the deepest layers of a trained model in order to check how the predictions are affected, providing an insight on the dynamics captured by each layer. To our knowledge, this is the first topology allowing for a direct observation of the behaviour encoded on each layer.

In Fig. 7, the same MMNIST sequences are predicted multiple times, removing a layer each time. The analysed model consists of 2 convolutional layers and 8 dGRU layers. Firstly, removing the last 2 dGRU layers has no significant impact on prediction. This shows that, for this dataset, the network has a

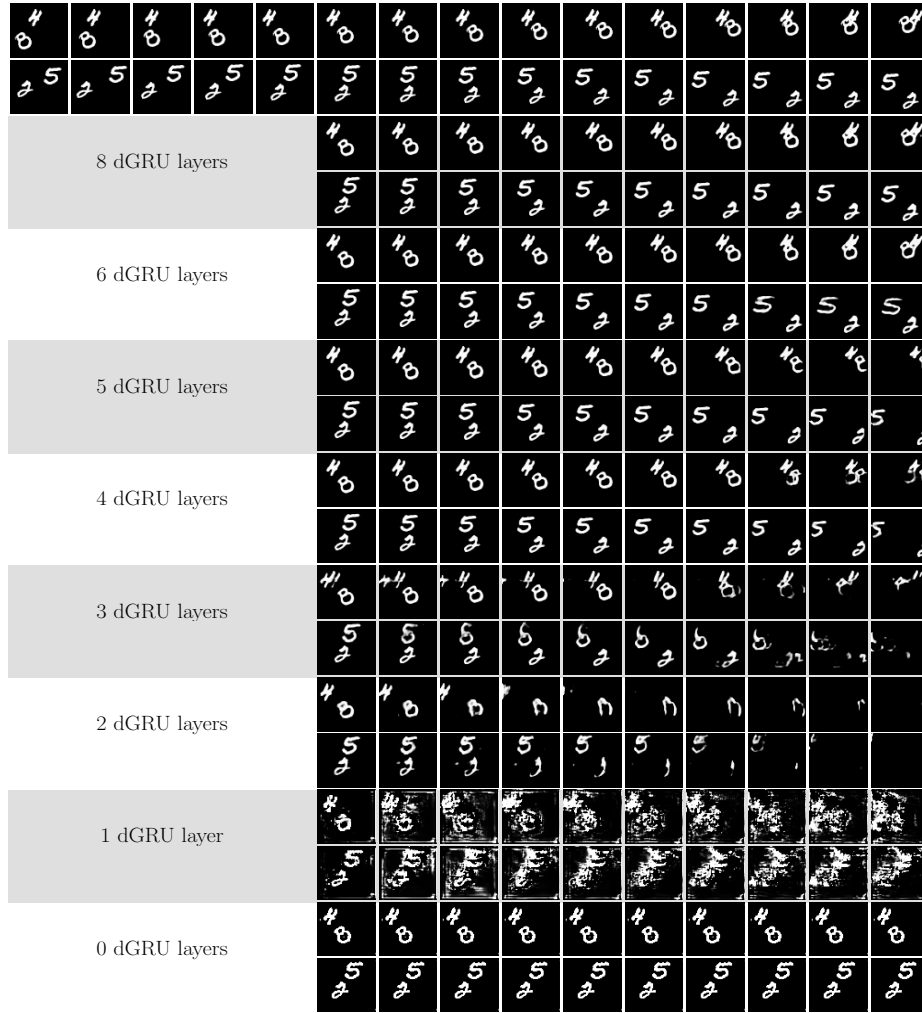


Fig. 7. Moving MNIST predictions with fRNN layer removal. Removing all dGRU layers (last row) leaves two convolutional layers and their transposed convolutions, providing an identity mapping.

higher capacity than required. Further removing layers results in a progressive loss of behaviours, from more complex to simpler ones. This means information at a given level of abstraction is not encoded into higher level layers. When removing the third deepest dGRU layer, the digits stop bouncing at the edges, exiting the image. This indicates this layer encodes information on bouncing dynamics. When removing the next one, digits stop behaving consistently at the edges: parts of the digit bounce while others keep the previous trajectory. While this also has to do with bouncing dynamics, the layer seems to be in charge of

recognising digits as single units following the same movement pattern. When removed, different segments of the digit are allowed to move as separate elements. Finally, with only 3-2 dGRU layers the digits are distorted in various ways. With only two layers left, the general linear dynamics are still captured by the model. By leaving a single dGRU layer, the linear dynamics are lost.

According to these results, the first two dGRU layers capture pixel-level movement dynamics. The next two aggregate the dynamics into single-trajectory components, preventing their distortion, and detect the collision of these components with image bounds. The fifth layer aggregates single-motion components into digits, forcing them to behave equally. This has the effect of preventing bounces, likely due to only one of the components reaching the edge of the image. The sixth dGRU layer provides coherent bouncing patterns for the digits.

5 Conclusions

We have presented Folded Recurrent Neural Networks, a new recurrent architecture for video prediction with lower computational and memory costs compared to equivalent recurrent AE models. This is achieved by using the proposed double-mapping GRUs, which horizontally pass information between the encoder and decoder. This eliminates the need for using the entire AE at any given step: only the encoder or decoder is executed for both input encoding and prediction, respectively. It also facilitates the convergence by naturally providing a noisy identity function during training. We evaluated our approach on three video datasets, outperforming state of the art techniques on MMNIST and UCF101, and obtaining competitive results on KTH with 2 and 3 times less memory usage and computational cost than the best scored approach. Qualitatively, the model can limit and recover from blur by preventing its propagation from low to high level dynamics. We also demonstrated stratification of the representation, topology optimisation, and model explainability through layer removal. Layers have been shown to successively introduce more complex behaviours: removing a layer eliminates its behaviours but leaves lower-level ones untouched.

Acknowledgements

The work of Marc Oliu is supported by the FI-DGR 2016 fellowship, granted by the Universities and Research Secretary of the Knowledge and Economy Department of the Generalitat de Catalunya. Also, the work of Javier Selva is supported by the APIF 2018 fellowship, granted by the Universitat de Barcelona. This work has been partially supported by the Spanish project TIN2016-74946-P (MINECO/FEDER, UE) and CERCA Programme / Generalitat de Catalunya. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPU used for this research.

References

1. Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R.H., Levine, S.: Stochastic variational video prediction. In: 6th International Conference on Learning Representations (2018)
2. Cricri, F., Honkala, M., Ni, X., Aksu, E., Gabbouj, M.: Video ladder networks. arXiv preprint arXiv:1612.01756 (2016)
3. Denton, E.L., Birodkar, v.: Unsupervised learning of disentangled representations from video. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems 30*, pp. 4417–4426. Curran Associates, Inc. (2017)
4. Ebert, F., Finn, C., Lee, A.X., Levine, S.: Self-supervised visual planning with temporal skip connections. arXiv preprint arXiv:1710.05268 (2017)
5. Finn, C., Goodfellow, I., Levine, S.: Unsupervised learning for physical interaction through video prediction. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) *Advances in Neural Information Processing Systems 29*. pp. 64–72. Curran Associates, Inc. (2016)
6. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (eds.) *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc. (2014)
7. Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) *Advances in Neural Information Processing Systems 29*. pp. 667–675. Curran Associates, Inc. (2016)
8. Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., Kavukcuoglu, K.: Video pixel networks. In: Precup, D., Teh, Y.W. (eds.) *Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research*, vol. 70, pp. 1771–1779. PMLR (2017)
9. Liang, X., Lee, L., Dai, W., Xing, E.P.: Dual motion gan for future-flow embedded video prediction. In: *Proceedings of the International Conference on Computer Vision*. pp. 1762–1770. IEEE, Curran Associates, Inc. (2017)
10. Liu, Z., Yeh, R., Tang, X., Liu, Y., Agarwala, A.: Video frame synthesis using deep voxel flow. In: *Proceedings of the International Conference on Computer Vision*. IEEE, Curran Associates, Inc. (2017). <https://doi.org/10.1109/ICCV.2017.478>
11. Lotter, W., Kreiman, G., Cox, D.: Deep predictive coding networks for video prediction and unsupervised learning. In: *International Conference on Learning Representations* (2016)
12. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. In: *International Conference on Learning Representations (ICLR)* (2016)
13. Michalski, V., Memisevic, R., Konda, K.: Modeling deep temporal dependencies with recurrent grammar cells. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems 27*, pp. 1925–1933. Curran Associates, Inc. (2014)
14. Oh, J., Guo, X., Lee, H., Lewis, R.L., Singh, S.: Action-Conditional Video Prediction using Deep Networks in Atari Games. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) *Advances in Neural Information Processing Systems 28*. pp. 2845–2853. Curran Associates, Inc. (2015)

15. Patraucean, V., Handa, A., Cipolla, R.: Spatio-temporal video autoencoder with differentiable memory. In: International Conference on Learning Representations Workshops (2015)
16. Prémont-Schwarz, I., Ilin, A., Hao, T., Rasmus, A., Boney, R., Valpola, H.: Recurrent ladder networks. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems* 30. pp. 6009–6019. Curran Associates, Inc. (2017)
17. Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., Chopra, S.: Video (language) modeling: a baseline for generative models of natural videos. arXiv preprint arXiv:1412.6604 (2014)
18. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local svm approach. In: Kittler, J., Petrou, M., Nixon, M.S. (eds.) *Proceedings of the 17th International Conference on Pattern Recognition*. vol. 3, pp. 32–36. IEEE (2004)
19. Sedaghat, N., Zolfaghari, M., Brox, T.: Hybrid learning of optical flow and next frame prediction to boost optical flow in the wild. arXiv preprint arXiv:1612.03777 (2016)
20. SHI, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.k., WOO, W.c.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) *Advances in Neural Information Processing Systems* 28, pp. 802–810. Curran Associates, Inc. (2015)
21. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
22. Srivastava, N., Mansimov, E., Salakhudinov, R.: Unsupervised learning of video representations using lstms. In: Bach, F., Blei, D. (eds.) *Proceedings of the 32nd International Conference on Machine Learning*. *Proceedings of Machine Learning Research*, vol. 37, pp. 843–852. PMLR (2015)
23. Van Amersfoort, J., Kannan, A., Ranzato, M., Szlam, A., Tran, D., Chintala, S.: Transformation-based models of video sequences. arXiv preprint arXiv:1701.08435 (2017)
24. Villegas, R., Yang, J., Hong, S., Lin, X., Lee, H.: Decomposing motion and content for natural video sequence prediction. In: *5th International Conference on Learning Representations* (2017)
25. Vukotić, V., Pinteá, S.L., Raymond, C., Gravier, G., Van Gemert, J.: One-step time-dependent future video frame prediction with a convolutional encoder-decoder neural network. In: *Netherlands Conference on Computer Vision* (2017)
26. Walker, J., Marino, K., Gupta, A., Hebert, M.: The pose knows: Video forecasting by generating pose futures. In: *Proceedings of the International Conference on Computer Vision*. pp. 3332–3341. IEEE, Curran Associates, Inc. (2017). <https://doi.org/10.1109/ICCV.2017.361>
27. Xue, T., Wu, J., Bouman, K., Freeman, B.: Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) *Advances in Neural Information Processing Systems* 29. pp. 91–99. Curran Associates, Inc. (2016)