Video Prediction on Moving MNIST

Edoardo Colonna / 1697124
Silverio Manganaro / 1817504

October 12, 2023

Abstract

In this report, we explain the work that has been carried out for the project of the exam of Deep Learning. We implemented and compared 3 different models for the task of Video Prediction on Moving MNIST[1]. Each of them is based on a different architecture, and the last one aims to extend one of the latest S.O.T.A. models for this task[2]. For each model we explain in detail the architecture, the main difficulties encountered, the solutions tried and the results obtained. These are shown both with images generated by the trained models and with MSE values, compared with S.O.T.A results from the literature. The repository of the project can be found here.

1 Introduction

Moving MNIST is a synthetic dataset consisting of two digits independently moving within the 64 × 64 grid and bouncing off the boundary. It is a standard and famous dataset used as a benchmark for spatiotemporal predicting learning. It is easier to work with since it has only one channel (images are black and white) and the frame's dimension is quite small.

The task of Video prediction consists of predicting subsequent frames given as input to the initial frames of a video. In particular, Moving MNIST is composed of a video of 20 frames of length. The more common version of the task provides that the first 10 frames are used as input, and then the output is 10 frames long, which are compared with the last 10 of the actual video to compute the error. To obtain such results, the models used should be able to abstract both the spatial context of the scene, so the position and the shape of the elements in the frame, and at the same time the temporal dimension that materializes in changes in position from frame to frame. The first approach to solving such a problem aimed to mix CNNs and RNNs, which are intrinsically capable of addressing the two natures of the task. Nowadays it has achieved a higher complexity in the models used, which can obtain incredible results. In the following paragraphs, we present three different methodologies that increase in complexity. After their explanation, are reported the results obtained and the conclusion about our work.

2 Methods

2.1 SimpleLSTM

The first model we implemented is the SimpleLSTM. Its architecture is composed of three main components: an Encoder, a Recurrent Neural Network and a Decoder. The Encoder and the Decoder

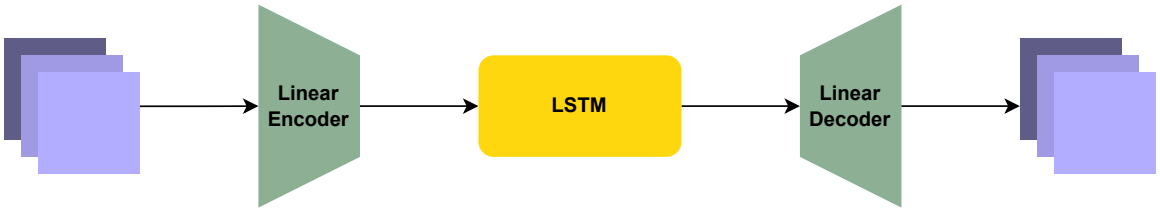

Figure 1: SimpleLSTM architecture diagram.

have a specular structure composed of a sequence of two dense layers, each followed by a ReLU activation function. A schematic picture of the model architecture can be seen in Figure 1.

Each frame in a sequence is initially processed by the Encoder as a flat one-dimensional vector to generate a lower-resolution representation of itself. This representation is then used as input to the RNNs, which in this case is an LSTM. It aims to create a hidden representation that takes into account the movement of the digits around the picture. When all the 10 input frames have been processed in this way, the output of the LSTM can be used iteratively by the LSTM itself to represent the next 9 frames of the predicted sequence. Lastly, the Decoder takes one by one these last 10 representations produced by the LSTM to reconstruct a frame having the same dimension as those given as input from each of them. Those frames are the model prediction which will be compared to the ground truth during the training phase through the Mean Squared Error.

2.2 ConvLSTM

The second model proposed is based on the ConvLSTM idea proposed in [3]. A ConvLSTM cell follows the same concept behind an LSTM one, which is a recurrent neural network with the addition of a forget gate, an input gate and an output gate to modulate what to maintain and what to discard of the input information signal of previous cells. The difference is that instead of having matrix multiplications as gates, in ConvLSTM they are substituted with convolution operators. This allows to mix in the same structure of the CNN ability to extrapolate hierarchical features from the frames with the RNN ability to remember information about previous frames.

In our model, we decided to use the original implementation of ConvLSTM as basic block for an Encoder-Decoder structure. Both the Encoder and the Decoder use 2 identical cells, for a total of 4 ConvLSTM cells. Before entering into the Encoder, the input passes through 2 convolutional layers, which shrink the dimension from **1x64x64** to $\mathbf{n_f}$**x32x32** where $\mathbf{n_f}$ is the number of hidden frames passed as hyperparameter. Lastly, the output of the Decoder passes through 2 de-convolutional layers, which bring it to the shape of the original frame, to compute the MSE. During the training phase, the first 10 frames of each video in the batch are passed as input into the encoder one after the other, without using the decoder since in this first part it only needs to accumulate information into the ConvLSTM cells. Then, for the other 10 frames that the model has to predict, the entire encoder-decoder structure is used, starting with the last real frame as input, and then using at each step the newly generated output as the input of the next step.

The initial tests with this configuration led us to poor results. The model was not able to learn, as we deduced from the very small decrease of training and validation error along the epochs. We tried every possible variation of the hyperparameters, but the results did not change. Since in the generated images, the shapes of the number digits, we decided to add skip layers to the architectures to maintain memories of the low-level features obtained as outputs of the first 2 convolutional layers. These two skip layers are then added to the input of the de-convolutional layers. This main change

Figure 2: ConvLSTM architecture diagram.

allowed us to obtain valuable results in the training phase. We tried different combinations of hyperparameters, activation layers and a little refining to discover which one gives the best results. The biggest improvement was obtained by changing the kernel size from **3x3** to **5x5**. The latter allows the network to capture more extensive information concerning the spatial changes in the scene. Since the frames of the MNIST dataset are not full of details and the digits are pretty grainy, using a larger kernel to detect movements turned out to be the right choice to capture movements. Would have been interesting, to validate deeply this intuition, to test higher kernel dimensions and also use more ConvLSTM cells together with different kernel sizes, but we haven't had a chance to try this yet because of the limitation of memory allowed by Google Colab platform, where we performed the trainings.

2.3 ConvTAU + Optical Flow

The third architecture we have considered is the ConvTAU [2] expanded with the usage of the optical flow as an additional encoding of the input frames. Figure 3 shows a schematic picture of the whole model.

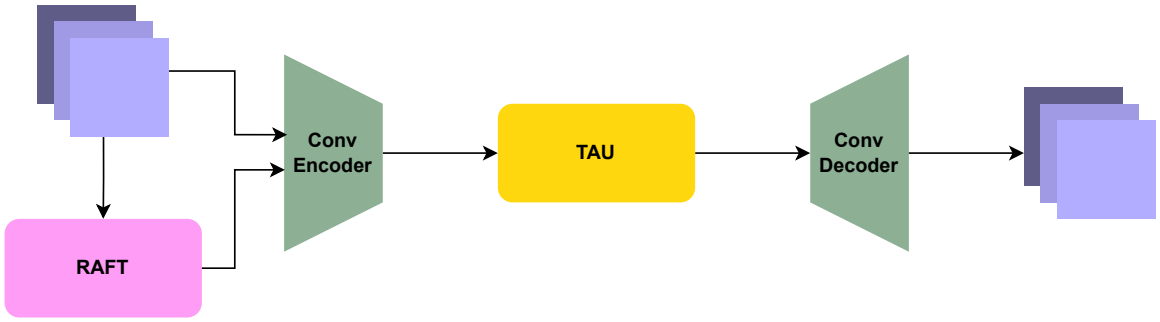

Figure 3: ConvTAU (with Optical Flow estimation) architecture diagram.

The input frames are firstly processed by an Optical Flow Estimator [4]. It is a pre-trained model of the RAFT deep architecture. Given two subsequent frames, it estimates the direction and the magnitude of the apparent velocity of each pixel of the current frame exploiting that at a previous instant. The result of an entire video sequence processing can be easily represented as a two-channel tensor.

The frames themselves and the optical flow estimation are then concatenated as single three-channel tensors and passed to the Encoder. The Encoder is made by a stack of N_S convolutional layers that progressively processes and downsamples the input frames. Each layer uses hid_S different

kernels to produce the encoded representation.

The hidden representation of the video frames is then passed to the TAU model, the core element of the architecture. It is composed of a stack of TAU modules and its aim consists of capturing the evolution of digits motion using a mixture of convolutional and linear operations. Each TAU module embodies the concept of temporal attention, so each processed sequence of frames is analyzed both concerning their independent content (statical attention) and the other frames in the series (dynamical attention). The primary objective of the statical attention is to capture intra-frame dependencies. It is accomplished via a concatenation of a set of three distinct convolutional layers to achieve an extensive receptive field. The first of them consists of a depth-wise convolutional layer which processes each input channel with a different kernel. Its output is then given as input to another depth-wise convolutional layer that performs also kernel dilation. The final statical attention is calculated through the last simple 1x1 convolutional layer. The main focus of the dynamical attention instead is to learn temporal evolutions of frame features. It is implemented simply as a fully connected network preceded by the average pooling operation. The general temporal attention is then the result of the product between the dynamical attention, the statical attention and the result of a 1x1 convolution applied to the input of the TAU module itself.

Lastly, the Decoder takes as input the output of the TAU model. This last module has a specular architecture concerning the Encoder since it is composed of a stack of N_S convolutional layers. In this case, however, it performs a progressive upsampling and a final convolution as a readout operation.

The outcome of the whole model is then a sequence of 10 predicted frames. The loss criterion used during the training consists of a weighted sum of the Mean Squared Error (which focuses on intra-frames differences) and the Kullback-Leibler divergence (which focuses on inter-frames variations) between the predicted and the ground truth frames.

To understand which hyperparameters influence the most the model performance, we have implemented a simple script to test some of those values. Finally, we have used the values which produced the best results on the same test set in the final ConvTAU model.

3 Results

We have trained each model for 5 epochs. This is a small amount of training steps, but it's enough to show clearly which model performs the best. Figure 4 shows their behaviour during the training phase.

We have compared the quality of the model outputs through the Mean Squared Error between the predicted and the ground truth frames. The results of the testing phase are shown in Table 1. To have a clearer idea of the meaning of those values, we have compared visually the outputs produced by our three models having as input the same video sequence. Figure 9 shows, for each model and each reference frame, the ground truth, the prediction and the difference between those two frames.

We have also trained for longer runs both the ConvLSTM and the original ConvTAU models to better understand how well they can become before overfitting. In Figure 5 are depicted the training and validation errors for the ConvLSTM model trained using 3x3 and 5x5 kernels. Is possible to notice the improvements in performance after having used larger kernels. After 20 epochs, the model performed an MSE of 1659 on the test set. We stopped at this number of epochs due to computation time limitation, but looking at both validation and test error, we can assume that the model would be able to still improve before reaching overfitting.

Comparing our extended ConvTAU architecture with the original version we can notice an im-

provement of about 7.7% on the results. This is due to the additional information added by the Optical Flow estimation.

All the data regarding the training sessions and the results we have achieved can be found also here.

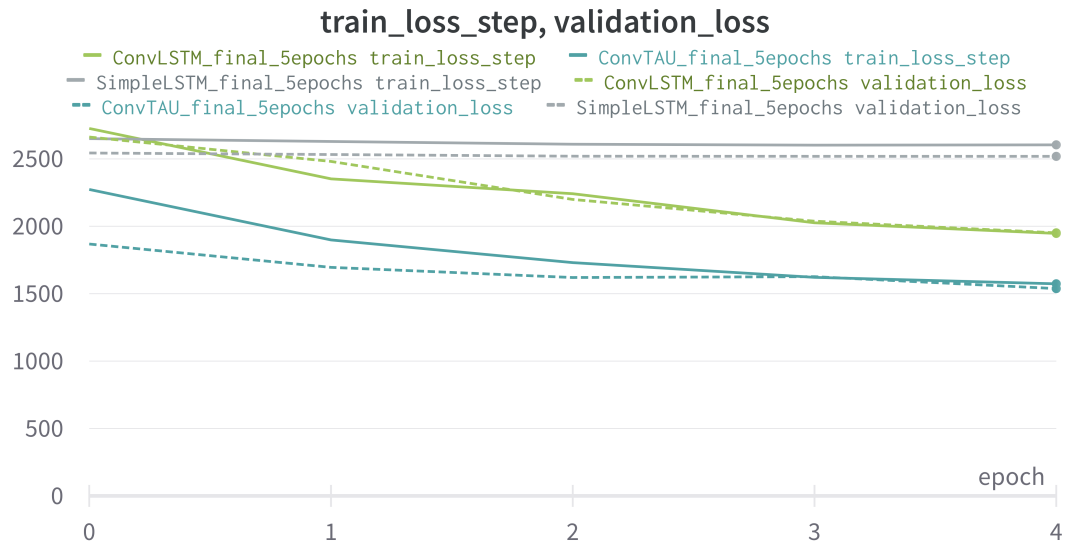

Figure 4: Training and Validation losses.

Figure 5: Training and Validation losses on ConvLSTM model with kernels 5x5 and 3x3.

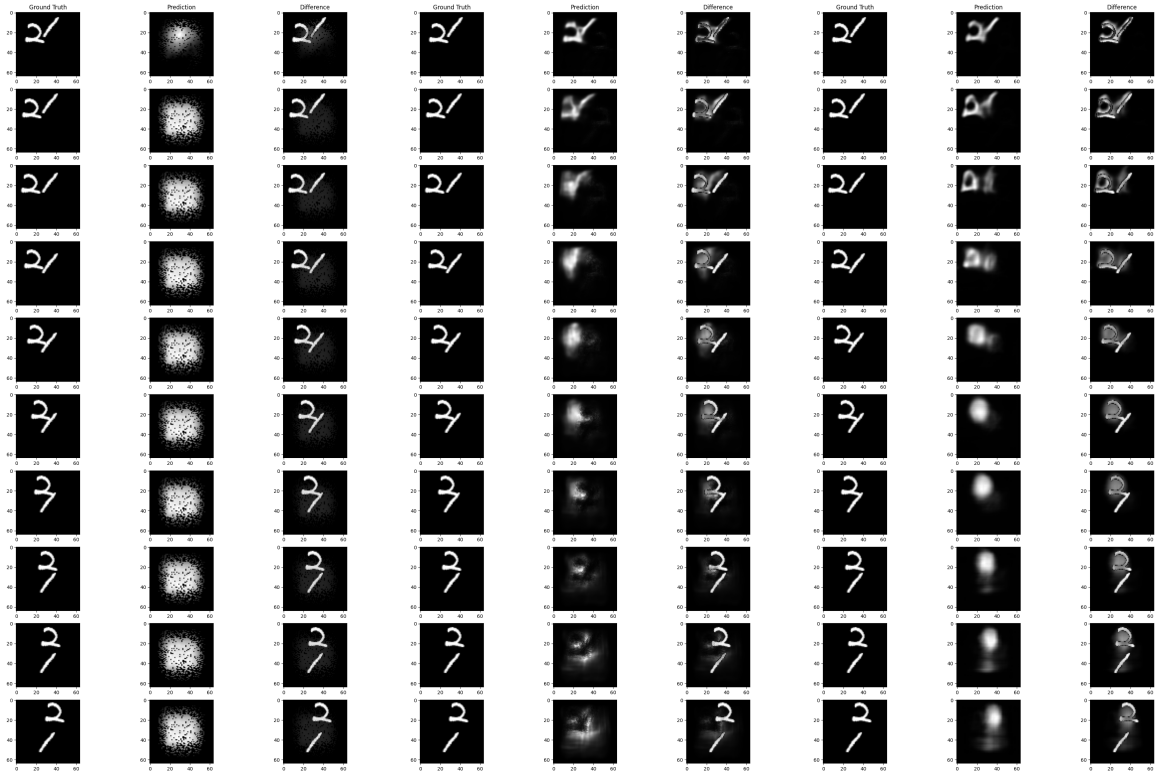

Figure 6: SimpleLSTM Figure 7: ConvLSTM Figure 8: ConvTAU

Figure 9: Output comparisons

Model	MSE
SimpleLSTM	2478
ConvLSTM	1927
ConvTAU original	1647
ConvTAU	**1520**

Table 1: MSE score after 5 epochs for each model

4 Conclusions

We have presented the work carried out for the Video Prediction on Moving MNIST task. We have presented and explained the 3 main models that we have used, along with the main difficulties encountered and the solutions that we tried to them. Then, we presented the MSE scores obtained over a small set of runs, showing also the generated frames of our models. Despite both our implementations and results being naive with respect to the current MSE score that S.O.T.A. models obtain, we managed to output results that show that our architectures can learn and improve on the given task. Of course, there is still much room for improvement, and in possible future works we would focus on applying some rules of thumb and actual practices which we tried in a first moment, when the models were not robust yet, and that, probably for this reason, did not bring in the hoped-for improvements. These include tests on weight initialization, specific optimizers, and sample and batch normalization. Beyond these practical ideas, would be also interesting to try conceptual approaches like teacher forcing or adversarial learning strategies as the one proposed in [5].

References

[1] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 843–852. [Online]. Available: https://proceedings.mlr.press/v37/srivastava15.html

[2] C. Tan, Z. Gao, L. Wu, Y. Xu, J. Xia, S. Li, and S. Z. Li, "Temporal attention unit: Towards efficient spatiotemporal predictive learning," 2023.

[3] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. kin Wong, and W. chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," 2015.

[4] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," 2020.

[5] M. e. a. Wang, "Recurrent adversarial video prediction network," 2022.