Predicting the dynamics of 2d objects with a deep residual network

François Fleuret
francois.fleuret@idiap.ch
Computer Vision and Learning group
Idiap Research Institute

Abstract

We investigate how a residual network can learn to predict the dynamics of interacting shapes purely as an image-to-image regression problem.

With a simple 2d physics simulator, we generate short sequences composed of rectangles put in motion by applying a pulling force at a point picked at random. The network is trained with a quadratic loss to predict the image of the resulting configuration, given the image of the starting configuration and an image indicating the point of grasping.

Experiments show that the network learns to predict accurately the resulting image, which implies in particular that (1) it segments rectangles as distinct components, (2) it infers which one contains the grasping point, (3) it models properly the dynamic of a single rectangle, including the torque, (4) it detects and handles collisions to some extent, and (5) it re-synthesizes properly the entire scene with displaced rectangles.

1 Problem definition

We implemented a simple 2d physics simulator to generate short sequences of interacting shapes. The simulation is quite crude but still includes an elastic collision model, a proper torque model, and (strong) fluid frictions proportional to velocity.

As illustrated with a few examples on Figure 1, each sequence is composed of gray-scale images of resolution 64×64, and is created as follows: We dispatch 10 rectangles of fixed size at random in the unit square, so that they do not overlap. Then we pick at random a point uniformly in the union of the rectangle interiors, and we apply there a constant force pulling upward for a constant time delay. This moves the grasped rectangle upward and may induce collisions with other rectangles, and make them move. The borders of the square area are impenetrable, hence rectangles grabbed near the top may have their motion constrained accordingly.

While the grasping point location is randomized for every sequence, the characteristics of the force and its duration are common to all the sequences.

From each such sequence, we produce three images: G_n, S_n, R_n which correspond, respectively, to: the grasping point image (all white, with a dot at the location of the grasp, as shown in the leftmost

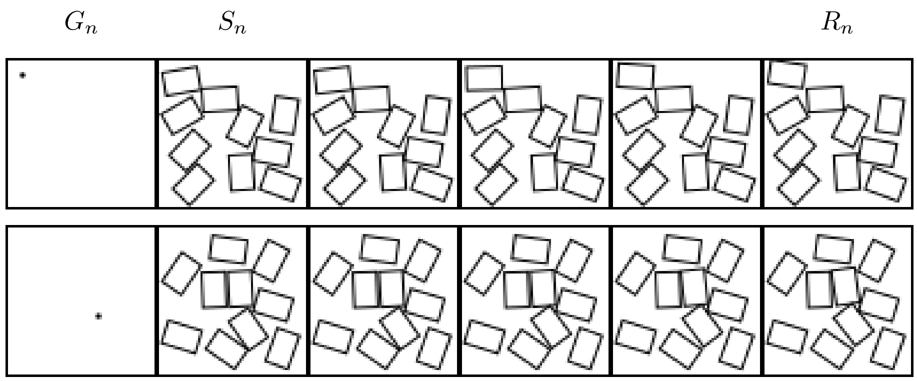

Figure 1: Each row corresponds to one sequence of our data-set. It is composed of six gray-scale images of size 64×64: the "grasping point image", followed by five frames. In each sequence, the rectangle originally containing the grasping point is pulled upward and moves accordingly. It may collide with and push other rectangles. We show several frames of each sequence for clarity here, but use only the two leftmost (S_n, G_n) and the rightmost image R_n of each sequence in our experiments, in which we try to predict the latter from the former.

column of Figure 1), the starting configuration, which is the first image of the sequence, and the resulting configuration, which is the last image of the sequence.

2 Network and training

We train a residual network [He et al., 2015] with 18 layer and 16 channels to predict R_n, given G_n and S_n as input. To ease the reading of long compositions of mappings, given two mappings f and g, let $f \, \triangleright \, g$ stand for $g \circ f$.

Our network follows the classical structure of the residual networks, and chains several identical modules of two convolutional layers. We define

- $\mathcal{C}_{c,d}$ a standard convolution layer [LeCun et al., 1998] with filters of size 5×5, padding of 2 to maintain the map size, c channels as input and d channels as output,
- $\mathcal{R}$ a ReLU rectifier layer [Glorot et al., 2011],
- $\mathcal{B}$ a batch-normalization layer [Ioffe and Szegedy, 2015],
- $\mathcal{I}$ the identity layer, and
- $\mathcal{M} = (\mathcal{C}_{q,q} \triangleright \mathcal{B} \triangleright \mathcal{R} \triangleright \mathcal{C}_{q,q} + \mathcal{I}) \triangleright \mathcal{B} \triangleright \mathcal{R}$ a two-layer resnet module [He et al., 2015] with the second batch normalization and non-linearity applied after summing the identity.

The structure of the full network is

$$\Psi = \mathcal{C}_{2,q} \triangleright \mathcal{B} \triangleright \mathcal{R} \triangleright \underbrace{\mathcal{M} \triangleright \ldots \triangleright \mathcal{M}}_{\times D} \triangleright \mathcal{C}_{q,1} \qquad (1)$$

with convolution filters of size 5×5, $q = 16$ channels in the internal encoding, and $D = 8$ resnet modules, each with two layers. It has a total of $104,417$ parameters, which is roughly $5^2 \times q^2 \times 2D$.

We minimize the quadratic loss between the predicted and the target training images, and train with $32,768$ samples. We use a standard stochastic gradient descent, randomizing the training set ordering for every epoch, using mini-batches of size 128, and a constant learning rate of 0.1. We did not tune the network structure, all the results obtained here are with the first attempt. A run with half the channels (i.e. $q = 8$) shows that it degrades noticeably the performance.

3 Results

We implemented the simulator in C++ and the network processing and performance evaluation in the Torch framework [Collobert et al., 2011]. The code is available under the GPL3.0 open-source license[1]

The loss decreases regularly, with no over-fitting. It is still going down after $2,000$ epochs, which takes slightly less than 30 hours on a NVIDIA gtx 1080, using the cuda toolkit 8.0, and cudnn 5.1.

3.1 Prediction

The resulting network makes an accurate prediction of the final configuration. We provide on Figure 2 five examples selected to illustrate the strengths and weaknesses of the prediction.

We observe that the network:

- detects the grasped rectangle, and moves it while keeping the other ones undisturbed if there is no collision.
- models translation and torque.
- propagates to some extent the dynamics when collisions occur (Figure 2(d)).
- models the hard borders around the area, although with some deformations (Figure 2(b)).
- implements the synthesis of the perturbed scene, which involves in particular the segmentation of the moving vs. non-moving parts, and synthesis of edges at multiple orientations.

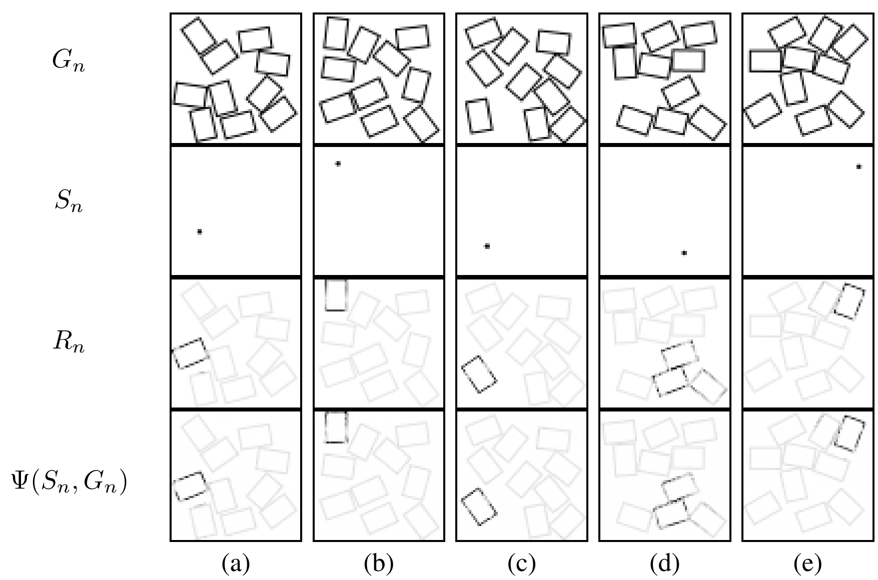

Figure 2: Some illustrative prediction results. Each column corresponds to an example. The first row shows the "grasping point" image G_n, the second row the starting configuration S_n, the third row the true resulting configuration R_n, and the last row the predicted resulting configuration $\Psi(S_n, G_n)$. **For clarity, we highlight the pixels in the two bottom rows proportionally to the difference with the starting configuration.** See § 3.1 for discussion.

[1] `https://gitlab.idiap.ch/francois.fleuret/dyncnn/`

3.2 Inner representation

To shade a light on the processing occurring in the network, we represent on Figure 3 the processing from top to bottom as the activations of the input layer, ReLU layers after each resnet module, and output layer for one of the examples.

We observe a homogeneity "per channel" (which translates here to "per column") probably because the resnet architecture favors processing near the identity, which results in gradual changes through layers, and discourages the shuffling of information across channels. As we can see, an important part of the computation aims at segmenting the grasped rectangle (channels 11 and 15), segmenting the moving rectangles (channels 4 and 5), and removing the moving parts (channels 1, 10, and 16).

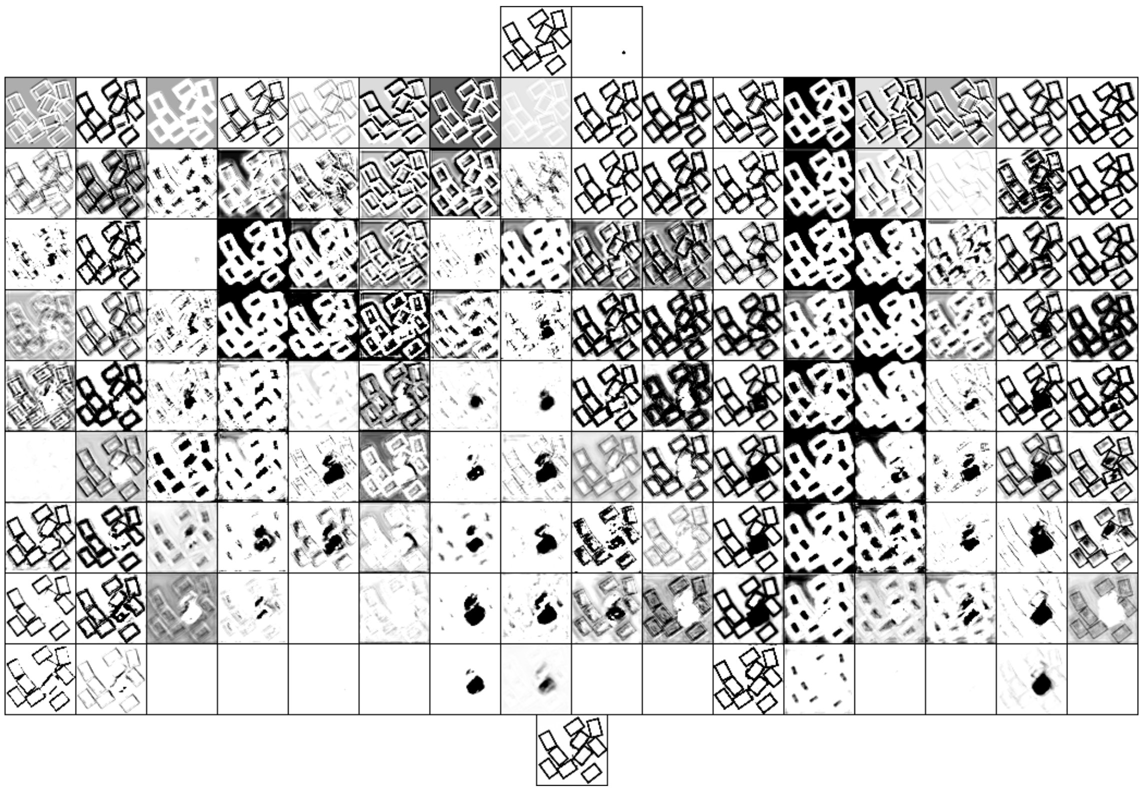

Figure 3: Activations in the input, internal ReLU, and output layers for the prediction pictured in Figure 2(d). See § 3.2 for discussion.

References

R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A Matlab-like environment for machine learning. In *Proceedings of the BigLearn NIPS Workshop*, 2011.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, volume 15, pages 315–323, 2011.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.