Convolutional Neural Networks for Movement Prediction in Videos

Alexander Warnecke, Timo Lüddecke, Florentin Wörgötter
III. Institute of Physics
Georg-August-University, Göttingen

Abstract

In this work we present a convolutional neural network-based (CNN) model that predicts future movements of ball given a series of images depicting the ball and its environment. For training and evaluation, we use artificially generated images sequences. Two scenarios are incorporated: Predicting in a simple table tennis environment and a more challenging squash environment. Classical 2D convolution layers are compared with 3D convolution layers that extract the motion information of the ball from contiguous frames. Moreover, we investigate whether networks with stereo visual input perform better than those with monocular vision only. Our experiments suggest that CNNs can indeed predict physical behaviour with small error rates on unseen data but the performance drops for very complex underlying movements.

1 Introduction

Predicting the movement of objects in images is a key capability in human perception and crucial for many tasks in everyday life. Hamrick et al.[4] provided evidence that human intuition for how dynamical systems evolve can be explained by the hypothesis that we use internal models of physics. It seems natural to ask whether a computer system is also able to build such an internal model for itself. This question is not only interesting from a theoretical point of view but also important for many applications in real life environments.

In this work, we consider two scenarios in a three-dimensional environment which we call table tennis and squash. The former refers to a flat surface with a bouncing ball while the latter involves a closed, cuboid-like room where multiple interactions with the wall are possible. The input is a fixed number of sequential images showing the movement of the ball. We develop a model that learns the trajectory of a ball from a fixed point in time, i.e. a sequence of (x, y, z) coordinates indicating the position of the ball in the future time steps. Since this is trivial in the empty space, in both of our scenarios, we put an emphasis on situations, where the ball changes its movement direction due to collisions with the surrounding walls. Naturally, this problem could also be solved analytically

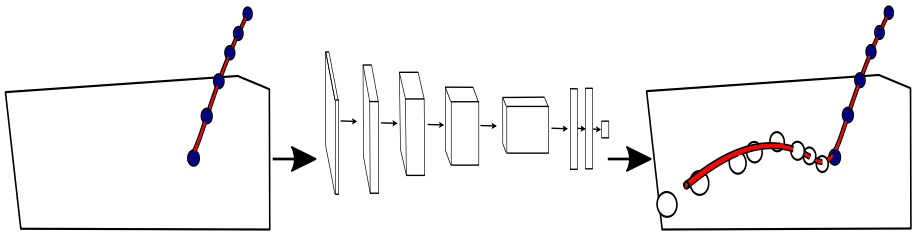

Figure 1: The movement of a ball is predicted by a convolutional neural network . It is shown the first six input frames and it predicts the trajectory of the future movement.

when all parameters are known (ball position, scene geometry, etc.). However, the task we consider here does explicitly require to estimate these parameters from the image. In order to do so, we apply convolutional neural networks, which have proven to be capable of understanding complex images.

2 Related Work

Neural networks have already been used to model physical phenomena in different setups. The approaches produce either images or numerical results to describe the movement. Michalski et al.[9] consider a rather abstract model that uses gated autoencoders which are turned into a recurrent neural network. The network takes a sequence of images $I_1, I_2, \ldots, I_t$ as input and learns transformations to create image I_{t+1} from the time series. The authors apply this model successfully to create movements like those of a ball bouncing on the floor. Lerer et al.[8] analyse the capability of convolutional neural networks to predict whether block towers will collapse. Different Convolutional neural networks are trained with rendered images and succeed in both, predicting whether the tower will collapse and creating an image of the final outcome. Walker et al.[13] predict the trajectory of each pixel from a static image for the next second with variational autoencoders using convolutional neural networks for the encoding and decoding part and achieve good results for images from challenging environments.

It is well known that simple movements of bodies are determined by equations of motion from Newtonian mechanics. Therefore, Wu et al.[14], Kyriazis et al.[11] and Bath et al.[2] propose models to estimate the parameters of these equations from images and videos to compute the dynamics. Mottaghi et al.[10] use a special convolutional neural network that takes a single image as input and predicts the dynamics of an object with two separate streams. One stream predicts in which newtonian scenario the scene is taking place and the other one generates features from the image while using an extra channel in the image indicating the object location. By merging these two information, the network can predict the trajectory of the moving object. However, tThese approaches

have two major drawbacks. Firstly, most humans cannot solve newtonian equations but can still predict the dynamics of a falling object, so these methods are not very natural. Secondly, they require pictures with an additional channel to indicate the position of the moving object.

Fragkiadaki et al.[3] investigate a setup in a 2D billiard environment. At some time t a convolutional neural network similar to the network in [7] receives the picture of the current situation, the last three preceding frames and the forces applied on the ball. The network predicts the movement, i.e. the velocity vector $(\Delta x, \Delta y)$ for each time step up to 20 following instants. The authors show that a model which was trained in a certain environment also performs well in unknown environments (different shape of the table, different length of the table walls). However, it still requieres a force information as an additional input to predict the movement which is an unhandy information to give.

Distinction from related work As in [3], we use convolutional neural networks for processing videos that show the movement of an object to predict the further motion. However, our environment is three dimensional which adds "depth" to the images and makes the task harder, moreover, we process videos instead of static images Another difference to [3] and [10] is that we use only the images as input without giving any other information about forces, momentum or position of the object. Moreover, the network in [3] looks only one time step into the future since it receives always the latest picture. Our model, in contrast, also predicts positions that are more than one time step away from the input frames. The output of our networks are, in contrast to images being predicted in [8], coordinates in $\mathbb{R}^3$ of the ball's future trajectory, which is a useful information for any system moving and acting in real world environments. To the best of our knowledge, there exists no approach only working on video input data generated from artificial three dimensional scenes predicting numerical information instead of images.

3 Scenarios and Data Acquisition

The video frames for training the convolutional neural network have been created using the open source 3D computer graphics program *blender* [1]. We use blender's physics engine to create rendered frames showing a realistic movement of a ball in different situations. We save the positions of the ball at every frame and train a convolutional neural network to predict the following coordinates after seeing the movement up to some timestep t. The first scenario is a table tennis scenario where a ball bounces on a table from different starting positions and in different directions. The second scenario is a squash environment where a ball is moving inside a box with multiple possible collisions from different starting positions.

[1] https://www.blender.org

3.1 Table tennis setup

The table and the ball for the table tennis setup have been created in real life proportions, i.e. the table has size 2.75m $\times$ 1.5m (length $\times$ width) and the ball has a diameter of 4cm. To increase computational performance and reduce the number of parameters, the frames were created using black-white images with a resolution of 128×128 pixels. The coordinate system is chosen such that the x axis runs along the longer side of the table and the y axis runs along the shorter side of the table. Consequently, the z axis indicates the height of an object "above" the table. The origin of the coordinate system is the middle point of the table and for computational stability, all axes are scaled by a factor of 10. To create random movements, the starting position of the ball is sampled according to a uniform distribution in the interval $[-5, 5]$ on the y axis and $[1.7, 4.5]$ on the z axis. The movement starts in the middle of the table, i.e. $x = 0$. Afterwards, we sample a random collision point on the table with $x \in [0, 13]$, $y \in [-7.5, 7.5]$ and $z = 0$. The ball flies from the starting point towards the collision point on the table, therefore the velocity and direction of the movement is defined and the physics engine can simulate the trajectory. The corresponding movement is recorded from two viewpoints, one at each corner of the table, to create a human-like viewpoint and give the network the chance to perceive depth in the frames.

3.2 Squash setup

The squash setup extend the table tennis setup to a bounded box to create complex movements with multiple collisions. It takes place in a box with 1m height and width and 0.7m length. The x and y axis run along the width and length of the box and the z axis along its height. Again, the origin is located in the middle of the box. Inside this box, a ball with a diameter of 4 cm is randomly positioned and "shot" into some direction in order to bounce against some walls of the box. The line of sight of the cameras runs along the y axis of the box. As before, the scene is recorded by two cameras, thus one side of the box is open such that the cameras can look into it. The starting positions were created by sampling x, y and z coordinates uniformly from the intervals $[-3.5, -1] \cup [1, 3.5]$, $[-4.5, 1]$ and $[-3.5, 3.5]$ respectively. To create the movement of the ball, we positioned a force field in blender randomly in front of the ball such that it is shot into the box. The coordinates of the force field are determined by adding random numbers Δx, Δy and Δz to the middle point of the sphere, where these numbers have been sampled uniformly from the intervals $[-0.5, 0.5]$, $[-0.3, -0.2]$ and $[-1, 1]$ respectively. Note that the y coordinate is shifted by a negative number in order to make the ball fly into the box and not towards the cameras. Finally, to make the setup harder and test the ability of the network to generalize, we shifted all 5 sides of the box randomly by a number sampled uniformly from the interval $[-1, 1]$ in order to create boxes of different size for every movement. The same way, we shifted the light, which is responsible for the shadow of the ball in the pictures, below

the top side of the box by random numbers from the interval $[-1, 1]$ in x and y direction. To take these difficult changes into account, the resolution of the videos is doubled to 256×256 pixels.

4 Trajectory Prediction Model

In this section we explain our CNN-based trajectory prediction model and describe various configurations in detail. In all cases a sequence of images is taken into consideration to predict future positions of the depicted ball in euclidean space.

4.1 Network architecture

2D vs. 3D convolution Firstly, we compare 2D convolutions with 3D convolutions. 2D convolution transforms an input of depth d into a two dimensional output, i.e. a feature map. 3D convolution filters have an additional temporal component and thus work in the spatio-temporal domain. By this construction, their output represents information from several contiguous frames and can thus capture motion information as shown in [6][5][12].

Input Representation Secondly, we compare different representations of the input to the network. Since we have six frames from each side of the table as input for the networks, we can either concatenate them and process them all at once or make use of the two different perspectives by feeding them separately into a network and merging them at some point. We will call the first model *concatenated input model* and the second one *two-stream model* in the following. Figure 2 shows the difference between the two models. While we can use 2D convolutions and 3D convolutions in the two-stream model, it is problematic to use 3D convolutions with the concatenated input because a 3D convolution filter is processing contiguous frames with the same weights at each position.

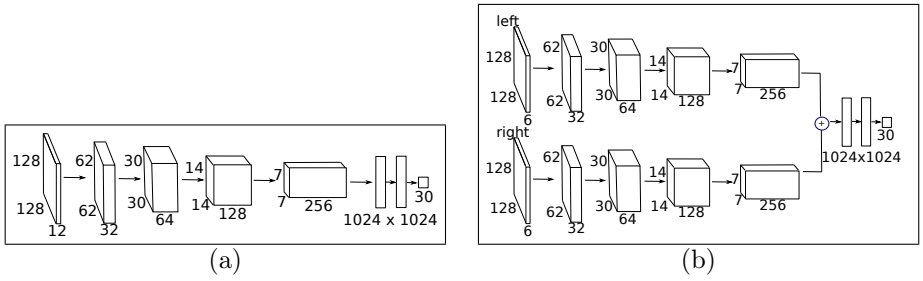

Figure 2: The different CNN models. (a): The concatenated input model processes all frames from each side at once. (b): The two-stream model processes the frames from the left and right side separately.

However, this would treat the transition between the (concatenated) perspectives in the same way as temporal transitions between two subsequent frames which is undesired. Hence 3D convolutions are only applied in the two-stream model.

Mono vs. Stereo Finally, we investigate the difference between mono- and stereo vision by using input of only one side in the concatenated input model. As shown in Figure 2 (a) the concatenated input network uses four convolution layers where each convolution is followed by batch normalization with $\epsilon = 0.001$, max pooling with a $(2, 2)$ filter and $RELU$ (rectified linear unit) where the size of the convolution filters is $(2, 2)$.

Network Head At the top of the network there are two fully connected layers where the first layer applies the RELU nonlinearity function and the second one a simple linear function, i.e. $f(x) = m \cdot x$ for some $m \in \mathbb{R}$, since we have to be able to predict negative coordinates. Between the two fully connected layers there is a dropout layer which randomly deactivates 50% of the units in the last layer during the learning process to prevent the network from overfitting. In the case of 3D convolutions, we use a network with two convolution layers with 10 and 20 feature maps respectively due to memory restrictions. The first layer uses a filter of size $(2, 2, 1)$ (width×height×time), i.e. the convolution is only performed in space and not in time so that the time axis is not collapsing too fast. The following filter uses a filter of size $(2, 2, 2)$, so that it convolves two contiguous frames in the time domain. As before, we apply batch normalization after convolution, as well as a 3D max-pooling with filter size $(2, 2, 1)$ for the first layer and $(2, 2, 2)$ for the second layer and RELU nonlinearity function. In each case, the streams are fused by using the sum of the feature maps of both streams.

Loss function We train the networks to predict the entire trajectory of a ball after some point in time t, i.e. a sequence $(\hat{x}_{t+1}, \hat{y}_{t+1}, \hat{z}_{t+1})$, $(\hat{x}_{t+2}, \hat{y}_{t+2}, \hat{z}_{t+2})$, $\ldots$, $(\hat{x}_{t+n}, \hat{y}_{t+n}, \hat{z}_{t+n})$ of coordinates, indicating the position of the ball in the future. For this work, we predict 8 future positions in the table tennis setup and 10 future positions of the ball in the squash setup for an input of 6 frames from each camera. This means, the output of the network is 30 dimensional in the squash setup and 24 dimensional in the table tennis scenario. Since it is very hard to have a good prediction for every point in time, we weighted our loss function as follows. The loss for a prediction sequence is defined by

$$L_\Theta = \sum_{i=1}^{10} w_{i-1}\big((x_{t+i} - \hat{x}_{t+i})^2 + (y_{t+i} - \hat{y}_{t+i})^2 + (z_{t+i} - \hat{z}_{t+i})^2\big), \qquad (1)$$

where $(x_{t+i}, y_{t+i}, z_{t+i})$ are the true positions of the ball, Θ is the set of all parameters of the network and the weights w_k are defined by

$$w_k = \exp(-k^{1/4}). \qquad (2)$$

Thus, the first weight is decaying exponentially but not too fast in order to give the last positions still enough weight in the loss function. If not mentioned differently, we present an average *error radius* in plots to describe the error. This is the average euclidean distance between the predicted position and the ground truth. The network was trained using the rmsprop algorithm [1], i.e.

$$w_{t+1,i} = w_{t,i} - \frac{\lambda}{\sqrt{H_{t,i} + \epsilon}} \cdot g_{t,i} \tag{3}$$

where $H_{t,i} = \gamma ||g_{t-1,i}||^2 + (1-\gamma)||g_{t,i}||^2$ is a running average over the gradients $g_{t,i} = \frac{\partial C}{\partial w_i}$ of the cost function C with respect to the parameter w_i at timestep t. For training, we used the parameters $\lambda = 1$, $\epsilon = 10^{-6}$ and $\gamma = 0.9$. The computations have been performed on a Geforce GTX TITAN X with batch size of 100 for 2D convolutions and 75 for 3D convolutions.

5 Results

In order to asses the performance of our model, we split the set of $50,000$ movements for both scenarios as follows: In the table tennis scenario, we removed movements from the training set which have their collision point with the table in a random area on the table, where the position of the area is chosen randomly but its size is fixed to contain about 5% of the movements for testing. This way, the test set contains only movements where the ball is moving to a direction the network has not seen before. Similarly, we removed 5% of the movements in the squash setup by choosing movements where the force field, which gives the ball its initial direction, was in a certain box relative to the starting point of the ball.

5.1 Table tennis scenario

Figure 3 (a) shows the average error radius of the networks (in cm) when predicting the 8 positions of the ball after the last input frame. We compare against the AlexNet architecture which was proposed by Krizhevsky et al. [7] for image classification without using pre-trained weights and replaced the softmax layer in the end with a fully connected layer with the right number of units to produce the trajectory coordinates. In comparison to the AlexNet, our networks use less layers, less feature maps and smaller convolution filters for processing the videos.

We can observe that all network architectures perform well for the first frames and produce error radii around 4cm which is the diameter of the ball and therefore a very good result. In the long term, we can see that the two-stream model with 3D convolutions is clearly the worst one, followed by the AlexNet and the network which has only monocular vision. The two networks with stereo vision show the best performance in the long term and especially the model with concatenated input frames performs well over all frame numbers.

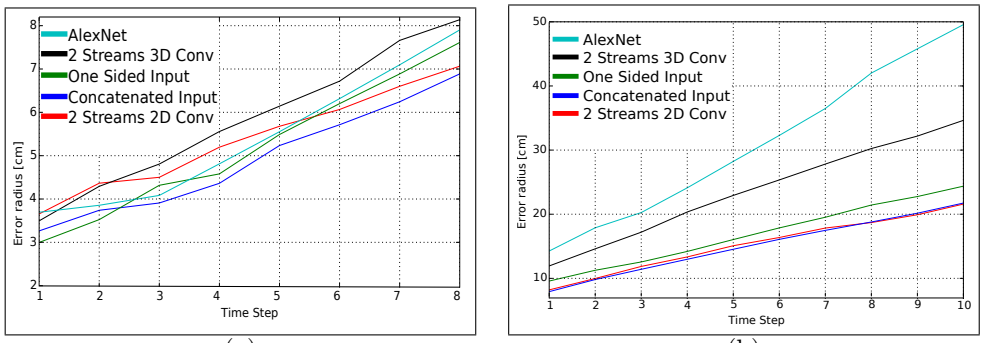

Figure 3: Error radius of the different models for the following frames in the tabel tennis setting (a) and the squash setting (b). Black: 2-Stream-3D-Convolution, Blue: Concatenated Input, Cyan: AlexNet, Green: One sided Input, Red: Two-Stream-2D-Convolution.

5.2 Squash scenario

The squash scenario contains more complex movements with multiple collisions and the environment in each trial varies with respect to the size of the box and the position of the light. The quantitative results shown in Figure 3 confirm that the task is harder for all the networks but at the same time we have a greater difference in the performance of the different models. In general, even the best models produce error radii of about 8 cm for the first frame and more than 20 cm for the last frame which is about 2.5 times greater than in the table tennis scenario. AlexNet and the two-stream-model with 3D convolutions are clearly outperformed by the other models and predict positions with a great error even for early frame numbers. Although the network with monocular vision only performs better than the former two, it is worse than the two networks with stereo image input which is the expected behavior but the difference between them is not too big. The best models are again the two-stream-model with 2D convolutions and the concatenated input model but compared to Figure 3 (a) the two models show an almost identical performance in this task indicating that it makes no difference whether the input is processed concatenated or separated in two streams. Table 1 shows the influence of the different dimensions on the error. We consider the average L^2 error in the blender coordinate system, which is the sum of the average squared error in the x, y and z dimension. We can see that the depth dimension y has clearly the highest error in all models and is roughly 20% higher for the network with monocular vision compared to the stereo vision ones.

Network	Δ_{L^2}	$\Delta_{L^2}(x)$	$\Delta_{L^2}(y)$	$\Delta_{L^2}(z)$
Concatenated Input	24.66	7.16	10.93	6.57
Two Stream 2D Convolution	25.12	6.49	10.92	7.71
One Sided Input	31.08	8.21	13.02	9.85

Table 1: Influence of the dimensions on the average L^2 error.

5.3 Qualitative Results

Figure 4 (a)-(d) show some predictions of the concatenated input model in the table tennis scenario where we can see that the network can predict different movements at different locations on the table. In general, the network performs best when the ball is flying towards the camera and when there is sufficient distance between two positions of the ball in the trajectory as seen in Figure 4 (a) and (b). Movements that are far away from the camera and have short trajectories like in Figure 4 (c) and (d) are more difficult for the network. Though it is interesting to see that the network is still predicting a trajectory that is meaningful for the movement which shows that it has really learned to predict trajectories during the learning process. Moreover, this result is comparable to the human performance in this task, i.e. the movements in pictures (c) and (d) would be hard to predict also for humans due to their distance to the point of view, especially when taking into account that the input images are only of size 128×128 pixels.
Figure 4 (e)-(j) show qualitative results created by the two-stream-network with 2D convolutions in the squash environment. Pictures (e)-(g) show good predictions of the network for straight movements. In comparison to the table tennis scenario, the movements are predicted before or after initial collisions with the walls took place and in boxes of different sizes. In picture (h) we can see a movement where the network has to predict two collisions and produces a rather big error. Figure 4 (e) - (j) show qualitative results created by the two-stream-network with 2D convolutions. pictures (e) - (g) show good predictions of the network for straight movements. In comparison to the table tennis scenario, the movements are predicted before or after initial collisions with the walls took place and in boxes of different sizes. In picture (h) we can see a movement where the network has to predict two collisions and produces a rather big error. Interestingly, we can clearly observe a knee in the predicted movement of the network indicating that it might has noticed that will be two changes in the direction of the ball. Pictures (i) and (j) show movements with the highest occurring error rates in the test set and show two important observations. Firstly, the network has really learned to predict movements since the predicted movements are reasonable trajectories of a flying ball. Secondly, the errors made by the network could also be made by humans. In Picture (i) one has to look very hard to see that the ball is bouncing against the back wall of the box and therefore changes its movement instead of just bouncing on the ground. Vice

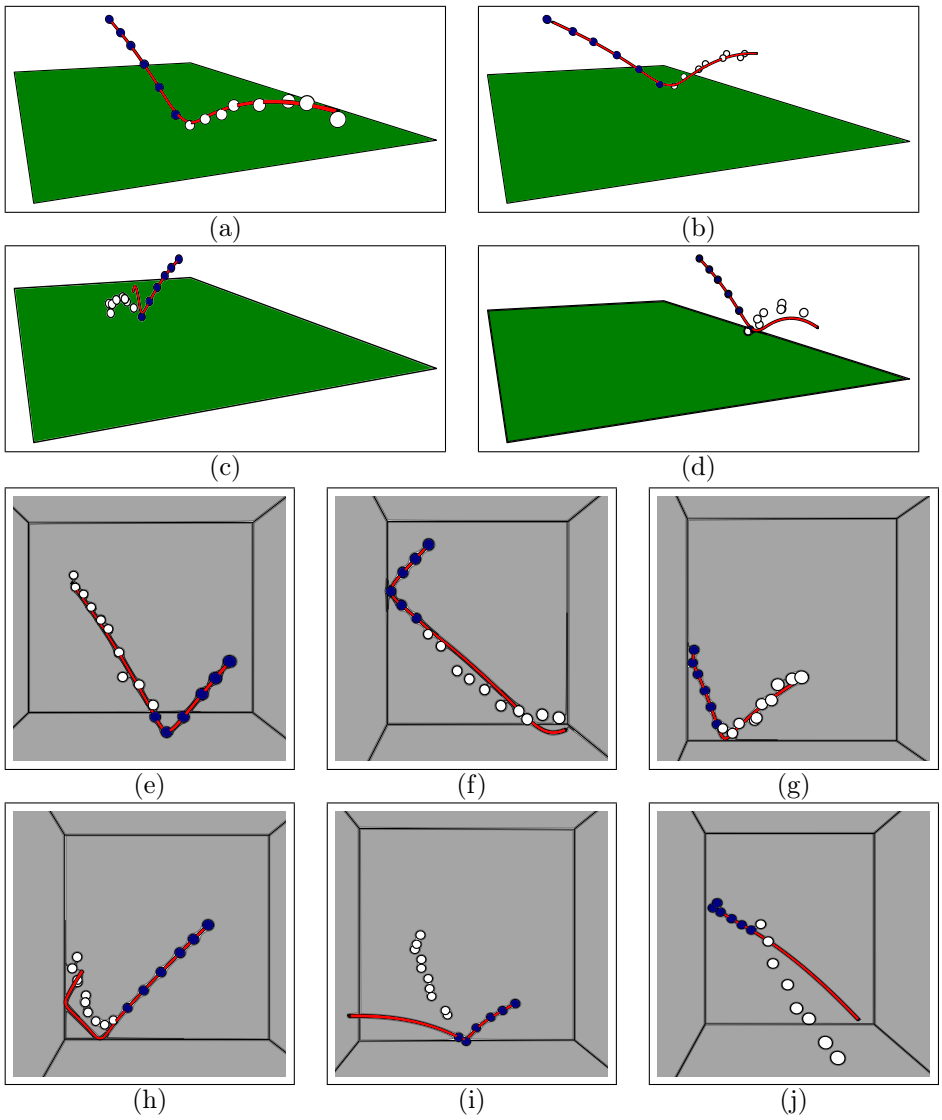

Figure 4: Qualitative results of the concatenated input model for the table tennis setup. Blue balls correspond to the input frames, white balls are predicted by the network and the red lines indicates the ground truth trajectory.

versa, in Picture (j) it is hard to say whether the ball will bounce against the back wall or not and the network assumed that it would and predicted a change in the movement direction.

6 Conclusion

We studied the ability of convolutional neural networks to build physical models to predict the movement of a ball in different scenarios for video input data that contains no further labels or information and with low resolution. Therefore, we compared network architectures with different kinds of convolution operation, monocular- and stereo visual input and different ways of processing stereo visual input. We found that these networks can predict many movements in changing environment and on unseen data with small error rates. Networks with stereo visual input perform better than those with monocular vision and can, just like human beings, gain advantage from the additional viewpoint for estimating the depth in images. 3D convolutions, however, are not able to gain advantage by processing contiguous frames of the movement at once and show clearly worse performance than networks with 2D convolutions only. All networks have problems when encountering difficult motions with multiple changes in directions and when the beginning of the movement is hard to see. However, often these falsely predicted movements still appear reasonable to a human.

References

[1] Neural networks for machine learning, lecture notes, `http://www.cs.toronto.edu/~tijmen/csc321`

[2] Bhat, K., Seitz, S., Popovic, J., Khosla, P.: Computing the physical parameters of rigid-body motion from video. In: Computer Vison - Eccv 2002, Pt 1. pp. 551–565. Springer-Verlag (2002)

[3] Fragkiadaki, K., Agrawal, P., Levine, S., Malik, J.: Learning visual predictive models of physics for playing billiards. CoRR abs/1511.07404 (2015)

[4] Hamrick, J.B., Battaglia, P., Tenenbaum, J.B.: Probabilistic internal physics models guide judgments about object dynamics. In: Proceedings of the 33th Annual Meeting of the Cognitive Science Society, CogSci 2011, Boston, Massachusetts, USA, July 20-23, 2011 (2011)

[5] Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. 35(1), 221–231 (Jan 2013)

[6] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1725–1732. CVPR '14 (2014)

[7] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012)

[8] Lerer, A., Gross, S., Fergus, R.: Learning physical intuition of block towers by example. In: Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016. pp. 430–438 (2016)

[9] Michalski, V., Memisevic, R., Konda, K.: Modeling deep temporal dependencies with recurrent grammar cells. In: In Advances in Neural Information Processing Systems 27. pp. 1925–1933. Curran Associates, Inc (2014)

[10] Mottaghi, R., Bagherinezhad, H., Rastegari, M., Farhadi, A.: Newtonian image understanding: Unfolding the dynamics of objects in static images. CoRR

[11] Nikolaos Kyriazis, I.O., Argyros, A.: Binding computer vision to physics based simulation: The case study of a bouncing ball. In: Proceedings of the British Machine Vision Conference. pp. 43.1–43.11. BMVA Press (2011)

[12] Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 4489–4497. IEEE (2015)

[13] Walker, J., Doersch, C., Gupta, A., Hebert, M.: An uncertain future: Forecasting from static images using variational autoencoders. CoRR abs/1606.07873 (2016)

[14] Wu, J., Yildirim, I., Lim, J.J., Freeman, B., Tenenbaum, J.: Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 127–135. Curran Associates, Inc. (2015)