
Learning A Physical Long-term Predictor

Sebastien Ehrhardt¹ Aron Monzpart² Niloy J. Mitra² Andrea Vedaldi¹

Abstract

Evolution has resulted in highly developed abilities in many natural intelligences to quickly and accurately predict mechanical phenomena. Humans have successfully developed laws of physics to abstract and model such mechanical phenomena. In the context of artificial intelligence, a recent line of work has focused on estimating physical parameters based on sensory data and use them in physical simulators to make long-term predictions. In contrast, we investigate the effectiveness of a single neural network for end-to-end long-term prediction of mechanical phenomena. Based on extensive evaluation, we demonstrate that such networks can outperform alternate approaches having even access to ground-truth physical simulators, especially when some physical parameters are unobserved or not known *a-priori*. Further, our network outputs a distribution of outcomes to capture the inherent uncertainty in the data. Our approach demonstrates for the first time the possibility of making actionable long-term predictions from sensor data *without* requiring to explicitly model the underlying physical laws.

1. Introduction

Most natural intelligences possess a remarkably accurate understanding of some of the physical properties of the world, as needed to navigate, prey, burrow, or perform any number of other ecologically-motivated activities. In particular, evolutionary pressure has caused most animals to develop the capability to perform fast and accurate predictions of mechanical phenomena. However, the nature of these mental models remains unclear and is being actively investigated (Hamrick et al., 2016).

¹University of Oxford, United Kingdom ²University College London, United Kingdom. Correspondence to: Sebastien Ehrhardt <hyenal@robots.ox.ac.uk>, Aron Monzpart <a.monzpart@cs.ucl.ac.uk>, Niloy J. Mitra <n.mitra@cs.ucl.ac.uk>, Andrea Vedaldi <vedaldi@robots.ox.ac.uk>.

Humans have developed an excellent *formal understanding* of physics; for example, at the level of granularity at which animals operate, mechanics is nearly perfectly described by Newton’s laws. However, while these laws are simple, their application to the description of a natural phenomena is anything but trivial. In fact, before such laws can be applied, a physical scenario needs first to be *abstracted* (e.g., by segmenting the world into rigid objects, describing those by mass volumes, estimating physical parameters). Then, except for the most trivial problems, predictions require the *numerical integration* of very complex systems of equations. It is therefore unclear whether animals would perform mechanical predictions in this manner.

In this paper, we investigate how an accurate understanding of mechanical phenomena can emerge in artificial systems, mimicking natural intelligence. Inspired by a number of recent works, we look in particular at how deep neural networks can be used to perform mechanical predictions in simple physical scenarios (Fig. 1). Among such prior works, by far the most popular approach is to use neural networks (Wu et al., 2016) to *extract from sensory data local predictions of physical parameters*, such as mass, velocity, or acceleration, that are then integrated by an external mechanism such as a *physical simulator* to obtain long term predictions. In other words, these approaches look at how an AI can abstract sensory data in physical parameters, but *not* how it can integrate such parameters over longer times. Further, such an approach assumes access to a simulator that accurately abstracts the physical world with appropriate Newtonian equations. Other attempts have also tried to replace the physical engine with a neural network (Battaglia et al., 2016) but did not really attempt to observe the physical world and deduce properties from it but rather to integrate the physical equations.

By contrast, in this paper we perform end-to-end prediction of mechanical phenomena with a single neural network, implicitly combining prediction and integration of physical parameters from sensory data. In other words, while most other approaches predict instantaneous parameters such as mass and velocity from a few video frames, our model directly performs long-term predictions of physical parameters such as position well beyond the initial observation interval. Thus, as our main contribution, we propose to do so by learning an internal representation of a physical sce-

nario which is induced by the observation of a few images and then is evolved in time by a recurrent architecture.

One of the challenges of extrapolating physical measurements is that the state of a physical system can be determined only up to a certain accuracy, and such uncertainty rapidly accumulates over time. Since no predictor can be expected to deterministically predict the future, predictors are best formulated as estimating *distribution of possible outcomes*. In this manner, predictors can properly account for uncertainty in the physical parameters, approximations in the model, or other limitations. Thus, our second contribution is to allow the neural network to explicitly model uncertainty by predicting distributions over physical parameters rather than their value directly.

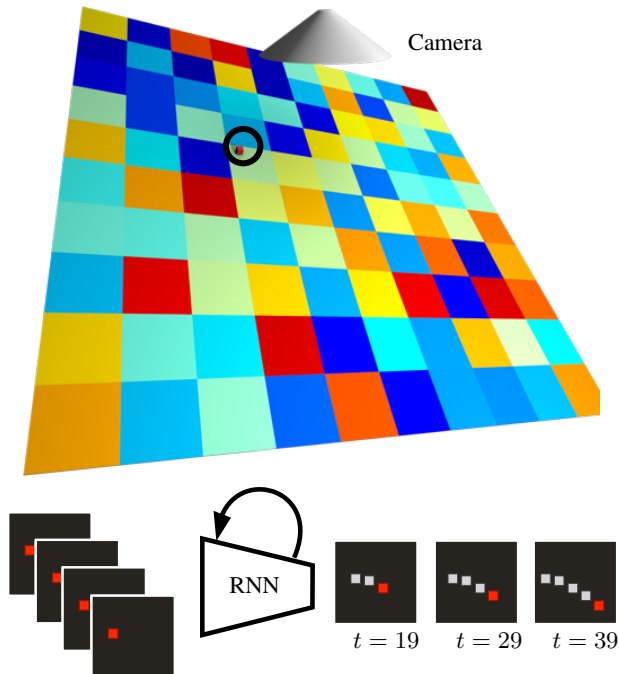
In our experiments, we let convolutional neural networks choose their own internal representation of physical laws. A soft prior is that convolutional architectures encourage learning structures that are local and spatially homogeneous, similar to the applicable physical laws that are also local and homogeneous. However, the network is never explicitly told what physical laws are. Our third contribution, therefore, is to look at whether such networks can learn physical properties that generalise beyond regimes observed during training.

The relation of our work to the literature is discussed in section 2. The detailed structure of the proposed neural networks is given and motivated in section 3. These networks are extensively evaluated on a large dataset of simulated physical experiments in section 4. A summary of our finding can be found in section 5.

2. Related Work

In this work we address the problem of long-term prediction of object positions in a physical environment without voluntary perturbation with an implicit learning of physical laws. Our work is closely related to a range of recent works in the machine learning community.

Learning intuitive physics. To the best of our knowledge (Battaglia et al., 2013) was the first approach to tackle intuitive physics with the aim to answer a set of intuitive questions (*e.g.*, will it fall?) using physical simulations. Their simulations, however used a sophisticated physics engine that incorporates prior knowledge about Newtonian physical equations. More recently (Mottaghi et al., 2016) also used static images and a graphic rendering engine (Blender) to predict movements and directions of forces from a single RGB image. Motivated by the recent success of deep learning for image processing (*e.g.*, (Krizhevsky et al., 2012; He et al., 2016)) they used a convolutional architecture to understand dynamics and forces acting behind



Input images $t = 0 \dots 3$

Figure 1: **MechaNets.** We consider the problem understanding and extrapolating mechanical phenomena with recurrent deep networks. An orthographic camera looks at a red cube sliding down a black slope with random inclination and heterogeneous friction coefficient (indicated in the top image by the fake coloured tiles). The camera observes the cube for four frames (bottom left) and a recurrent network (bottom middle) predicts the long term motion for up to 40 frames (bottom right). Our goal is to investigate to which extent recurrent networks can develop an internal representation of physics.

the scenes from a static image and produced a “most likely motion” rendered from a graphics engine. In a different framework (Lerer et al., 2016) and (Li et al., 2016) also used the power of deep learning to extract an abstract representation of the concept of stability of block towers purely from images. These approaches successfully demonstrated that not only was a network able to accurately predict the stability of the block tower but in addition, it could identify the source of the instability. Other approaches such as (Agrawal et al., 2016) or (Denil et al., 2016) also attempted to learn intuitive physics of objects through manipulation. None of these approaches did, however, attempt to precisely model the evolution of the physical world.

Learning dynamics. Learning the evolution of an object’s position also implies to learn about the object’s dynamics regardless of any physical equations. While most successful techniques used LSTM-s (Hochreiter & Schmidhuber, 1997), recent approaches show that propa-

gation can also be done using a single cross-convolution kernel. The idea was further developed in (Xue et al., 2016) in order to generate a next possible image frame from a single static input image. The concept has been shown to have promising performance regarding longer term predictions on the moving MNIST dataset in (De Brabandere et al., 2016). The work of (Ondruska & Posner, 2016) also shows that an internal hidden state can be propagated through time using a simple deep recurrent architecture. These results motivated us to propagate tensor based state representations instead of a single vector representation using a series of convolutions. In the future we also aim to experiment with approaches inspired by (Xue et al., 2016).

Learning physics. Works of (Wu et al., 2015) and its extension (Wu et al., 2016) propose methods to learn physical properties of scenes and objects. However in (Wu et al., 2015) the MCMC sampling based approach assumes a complete knowledge of the physical equations to estimate the correct physical parameters. In (Wu et al., 2016) deep learning has been used more extensively to replace the MCMC based sampling but this work also employs an explicit encoding and computation of physical laws to regress the output of their tracker. (Stewart & Ermon, 2016) also used physical laws to predict the movement of a pillow from unlabelled data though their approach was only applied to a fixed number of frames.

In another related approach (Fragkiadaki et al., 2015) attempted to build an internal representation of the physical world. Using a billiard board with an external simulator they built a network which observing four frames and an applied force, was able to predict the 20 next object velocities. Generalization in this work was made using an LSTM in the intermediate representations. The process can be interpreted as iterative since frame generation is made to provide new inputs to the network. This can also be seen as a regularization process to avoid the internal representation of dynamics to decay over time which is different to our approach in which we try to build a stronger internal representation that will attempt to avoid such decay.

Other research attempted to abstract the physics engine enforcing the laws of physics as neural network models. (Battaglia et al., 2016) and (Chang et al., 2016) were able to produce accurate estimations of the next state of the world. Although the results look plausible and promising, long term predictions are still an issue in such frameworks. Note, that their process is an iterative one as opposed to ours, which propagates an internal state of the world through time.

Approximate physics with realistic output. Other approaches also focused on learning the production of realistic future scenarios ((Tompson et al., 2016) and (Jeong

et al., 2015)), or inferring collision parameters from monocular videos (Monzspart et al., 2016). In these approaches the authors used physics based losses to produce visually plausible yet erroneous results. They however show promising results and constructed new losses taking into account additional physical parameters other than velocity.

3. Mechanics Networks

In this section, we introduce a number of neural network models that can predict the behaviour of a simple mechanical system. We start by describing the physical setup and then we introduce the proposed network architectures.

3.1. Physical setup

The physical setup (Fig. 1) consists of a small object sliding down an inclined plane. For notational simplicity, we identify the 3D Euclidean space with the underlying vector space \mathbb{R}^3 and denote as $\mathbf{p} = (p_x, p_y, p_z) \in \mathbb{R}^3$ the coordinates of points as well as of vectors. The plane, which for simplicity passes through the origin, has equation $\pi = \{\mathbf{p} \in \mathbb{R}^3 : \langle \mathbf{n}, \mathbf{p} \rangle = 0\}$, where \mathbf{n} is the plane unit normal vector. In addition to the normal \mathbf{n} , capturing the inclination, the plane has also a Coulomb *friction coefficient* ρ , which in the simple case is *homogeneous*, but which can also be a spatially varying quantity $\rho : \pi \rightarrow \mathbb{R}_+$.

We set the camera to be located above the plane, at height $h > 0$, centered at point $(0, 0, h)$, and looking downwards along the Z-direction $(0, 0, -1)$. The camera axes are aligned to the world axes and the camera projection model is orthographic; in this setting, a world point \mathbf{p} simply projects to pixel (p_x, p_y) in the image. Note that h does not have an influence on the generated image and can be dropped.

The sliding object is a cube with center of mass $\mathbf{q}(t) = (q_x(t), q_y(t), q_z(t))$ at time t , which projects to pixel $\mathbf{y}(t) = \alpha(q_x(t), q_y(t)) + \beta$ in the image. Here we consider $H_i \times W_i = 128 \times 128$ images with pixels of size $\alpha = 1$ and offset $\beta = (-64, -64)$. Initially, the cube is placed at rest on top of the plane at a random location (q_x^0, q_y^0) , and then starts to slide under the effect of gravity. The cube motion is also affected by friction.

An experiment instance is a tuple $\alpha = (q_x^0, q_y^0, \mathbf{n}, \rho)$, consisting of the values of the initial object position, the plane normal, and the friction coefficient or distribution. The inclination \mathbf{n} is arbitrary (within limits), such that the object can slide in any direction. These parameters, as well as many other constant parameters described in section 4.1, are passed to a physical simulator and rendered to simulate the experiment, resulting in a sequence of color images $\mathcal{X}_T^\alpha = (\mathbf{x}^\alpha(0), \mathbf{x}^\alpha(1), \dots, \mathbf{x}^\alpha(T-1))$. The simulator

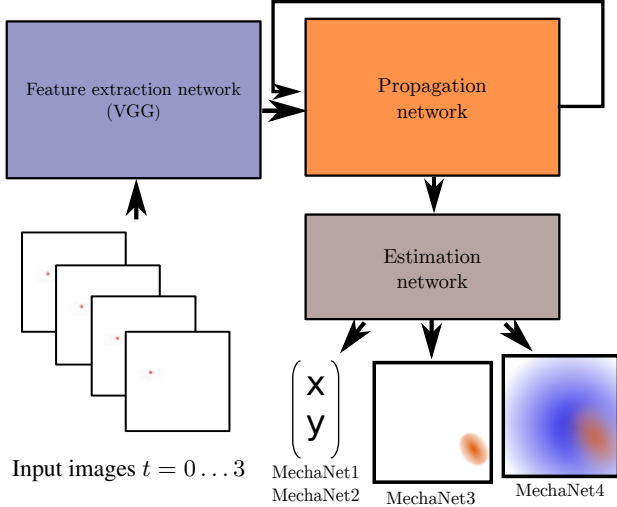


Figure 2: Overview of our proposed pipeline. The first four images of a sequence first pass through a partially pre-trained feature extraction network to build the concept of physical state. It then recursively passes through a propagation layer to produce long-term predictions about the future positions of the objects. Extrapolation requires us to handle the notion of uncertainty, which is why *MechaNet4* performs the best under changing physical conditions in Scenario S2, see Table 2.

also produces the ground-truth center of mass projections $\mathcal{Y}_T^\alpha = (\mathbf{y}^\alpha(0), \dots, \mathbf{y}^\alpha(T-1))$. Note that, for the purpose of learning predictors, physics needs not to be specified further; while in fact a complete set of parameters are required to run the physical simulation, the predictor learns automatically to extract the required information from the observed images.

3.2. Neural network architectures

We focus on long-term predictors $\Phi: \mathcal{X}_{T_0} \mapsto \mathcal{Y}_T$ that take as input the first $T_0 = 4$ frames \mathcal{X}_{T_0} of a video sequence \mathcal{X}_T and produce as output a long-term estimate \mathcal{Y}_T of the location of the object’s center of mass at times $t = 0, 1, \dots, T$, where $T \gg T_0$.

Our method comprises **three building blocks** (Fig. 2): a feature extractor, a propagation network and an estimation network. The core of our model is the internal representation of the physics, initialized by the feature extractor, updated by the propagation module, and decoded by the estimation module. We compare two representation types: a *vector representation*, in which each frame is encoded as C -dimensional vector (or $1 \times 1 \times C$ tensor), and a $H \times W \times C$ *tensor representation*. The importance of this difference is that the vector representation is spatially *concentrated*, whereas the tensor representation is spatially

distributed.

Next, the three modules are discussed in detail.

(i) Feature extraction network. The predictor $\mathcal{Y}_T = \Phi(\mathcal{X}_{T_0})$ starts by extracting information from T_0 video frames. Similarly to (Fragkiadaki et al., 2015), the RGB channels of the images are concatenated in a single $H_i \times W_i \times 3T_0$ tensor and this is processed by a convolutional neural network ϕ_{init} , obtaining a $\phi_{\text{init}}(\mathbf{x}_0, \dots, \mathbf{x}_{T_0-1}) \in \mathbb{R}^{H \times W \times C}$ tensor output. These features serve as the internal physical representation of our network that is propagated through time. Inspired by (Fragkiadaki et al., 2015), we start from the VGG16 network pre-trained on ImageNet (Simonyan & Zisserman, 2015). The network is cut and the last layer adapted as needed. In particular, starting from a $(H_i, W_i) = (128, 128, 3)$ image, the vector representation uses the $(H, W, C) = (1, 1, 128)$ dimensional output of layer `fc6`, and the tensor representation uses the $(8, 8, 512)$ dimensional output of `conv5` instead. All feature extraction layers are frozen in training, except the new layer `fc6` and `conv1`, whose shape changes.

(ii) Propagation network. The internal representation initialized by the feature extractor is evolved through time by the propagation network $F: \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{H \times W \times C}$. Formally, the internal state S_t is initialized as $S_0 = \phi_{\text{init}}(\mathcal{X}_{T_0})$ and updated by iteratively applying F as $S_{t+1} = F(S_t) = F^t(\phi_{\text{init}}(\mathcal{X}_{T_0}))$ for $t \geq 1$ (note that there is an index shift between state and time, so that S_t predicts the object position at time $t + T_0 - 1$). For the vector representation, the propagation network is an LSTM with 128 hidden units. Since there are no more observations after T_0 , the LSTM input at time t is set to the internal state of the LSTM at the previous time. This is similar in approach to (Cho et al., 2014), although our output is directly fed to the network without re-embedding. For the distributed representation, we use a simple chain of two convolution layers (with 256 and 512 filters respectively, of size 3×3 , stride 1, and padding 1) interleaved by a ReLU layer. When using *discrete probability map*, the representation S_t is normalised channel-wise in L^2 norm after each update in order to avoid the decay of intermediate propagation layers.

(iii) Estimation network. In the simplest instance, the network predictor estimates directly the values $\hat{\mathcal{Y}}_T = (\hat{\mathbf{y}}(0), \dots, \hat{\mathbf{y}}(T-1))$ of the object’s center of mass $\hat{\mathbf{y}}(t) \in \mathbb{R}^2$ during the sequence. The learning loss is simply the average squared distance between predicted and ground-truth locations:

$$\mathcal{L}(\hat{\mathcal{Y}}_T, \mathcal{Y}_T) = \frac{1}{T} \sum_{t=0}^{T-1} \|\hat{\mathbf{y}}(t) - \mathbf{y}(t)\|^2.$$

As discussed above, however, it is preferable to predict the *uncertainty* of the estimate as well. While in some cases

this cannot improve accuracy directly (i.e in the bivariate gaussian case), it is interesting to see if a network is able to develop an internal sense of prediction errors. Further, probabilistic modelling may help the network discount difficult-to-predict points during training, which may otherwise work as outliers negatively affecting training.

We propose to do so in two ways. In the first approach, we predict the mean and variance $\hat{\mathcal{Y}}_T = (\mu(t), \Sigma(t); t = 0, \dots, T - 1)$ of a bivariate Gaussian distribution $\mathcal{N}(\cdot; \mu, \Sigma)$. The loss is the negative log-likelihood of the measured object locations:

$$\mathcal{L}_{\text{nm}}(\hat{\mathcal{Y}}_T, \mathcal{Y}_T) = -\frac{1}{T} \sum_{t=0}^{T-1} \log \mathcal{N}(\mathbf{y}(t); \mu(t), \Sigma(t)).$$

In practice, the neural network estimates the two dimensional vector $\mu(t)$ as well as a three dimensional vector $\lambda_1(t), \lambda_2(t), \theta(t)$ with the first two being the eigenvalues of $\Sigma(t)$, and the third entry being the angle of the rotation matrix in the decomposition $\Sigma(t) = R(-\theta(t)) \begin{bmatrix} \lambda_1(t) & 0 \\ 0 & \lambda_2(t) \end{bmatrix} R(\theta(t))$. In order to ensure numerical stability, eigenvalues are constrained to be in the range $[0.01 \dots 100]$ by setting them as the output of a scaled and translated sigmoid $\lambda_i(t) = \sigma_{\lambda, \alpha}(\beta_i(t))$, where $\sigma_{\lambda, \alpha}(z) = \lambda / (1 + \exp(-z)) + \alpha$. For more details regarding the training procedure of this model please see section 4.3.

In the second approach, we predict *discrete probability maps* $\mathcal{Y}_T = (p(0), \dots, p(T - 1))$, where $p(t) \in \mathbb{R}^{H_i \times W_i}$ and $p(t)_{ij}$ is the probability that the object’s center of mass is contained in a 1×1 square centered at location $(j - W_i/2, i - H_i/2)$. Similar to the Gaussian loss, we use the negative log-likelihood of the ground-truth observations as loss:

$$\mathcal{L}_{\text{heat}}(\hat{\mathcal{Y}}_T, \mathcal{Y}_T) = 2 \log \delta - \frac{1}{T} \sum_{t=0}^{T-1} \log p(t)_{\lfloor \mathbf{y}(t) \rfloor},$$

where $\lfloor \cdot \rfloor$ is the rounding operator and $\delta = 1$ is the sampling step.¹ The probability maps $p(t)$ sum to one and are obtained by applying the softmax operator to a tensor $A(t) \in \mathbb{R}^{H_i \times W_i}$ estimated by the neural network:

$$p(t)_{ij} = \frac{e^{A(t)_{ij}}}{\sum_{mn} e^{A(t)_{mn}}}.$$

All the output predictions at time $t + T_0 - 1$ are extracted from the internal state S_t by a single layer $L(S_t)$. The layer L is linear and fully-connected layer for \mathcal{L} and \mathcal{L}_{nm} , and a deconvolutional layer similar to (Long et al., 2015) in the

case of $\mathcal{L}_{\text{heat}}$. Outputs at times $t = 0, 1, \dots, T_0$ are all predicted from S_0 using an analogous but independently-trained fully-connected layer L' . Overall, the output of the predictor is given by:

$$\Phi(\mathcal{X}_{T_0}) = (L'(S_0)_1, \dots, L'(S_0)_{T_0}, L(S_1), \dots, L(S_{T-T_0-1})).$$

4. Experiments

4.1. Data generation

Experiments consider three variants of the physical setup described in section 3.1, called Scenarios S0, S1, and S2. Different scenarios sample experiments $\alpha = (q_x^0, q_y^0, \mathbf{n}, \rho)$ of increasing difficulty. The parameters of each scenario are summarised in Table 1 and described next. The plane normal \mathbf{n} was obtained by rotating the Z axis around the X and Y axis by random angles θ_x, θ_y (Scenario S0 uses a fixed inclination). For Scenarios S1 and S2, the Coulomb friction coefficient ρ of the plane is homogeneous and sampled uniformly at random. For Scenario S2, the plane is split in 10×10 patches, each with a random friction coefficient sampled independently. The friction upper bound was chosen so that the object always slides along the slope.

Scenario	\mathbf{n} rotation	ρ
S0	$\theta_x = 0, \theta_y = \frac{\pi}{6}$	$\rho \in \mathcal{U}(10^{-4}, 10^{-1})$
S1	$\theta_x, \theta_y \in \mathcal{U}(-\frac{\pi}{6}, \frac{\pi}{6})$	$\rho \in \mathcal{U}(10^{-4}, 10^{-1})$
S2	$\theta_x, \theta_y \in \mathcal{U}(-\frac{\pi}{6}, \frac{\pi}{6})$	$\forall_{i,j=1\dots 10} \rho_{i,j} \in \mathcal{U}(0.5, 5)$

Table 1: Data generation parameters.

The plane was rendered as a black object so that no static visual cues allow deducing any of the physical parameters except the initial position of the cube; instead the predictor has to approximate physics as needed by observing the motion of the object during the first $T_0 = 4$ frames of each experiment.

Each experiment was run for at most 240 frames, or terminated early if the object left the field of view. In order to observe enough movement in each recorded sequence, the first 30 video frames of each video were removed, and the rest of the video was sub-sampled by a factor of 3. In practice, most experiments consist of 40-50 images.

The dataset contains 12,500 experiments for each Scenario, 70% of which are used for training, 15% for validation, and 15% for test.

Implementation details. The object’s starting position is initialized randomly using rejection sampling in such a way that (q_x^0, q_y^0) falls in the slope quadrant that contains the largest visible h coordinate. This procedure generates samples that have most of their trajectory visible to the camera.

¹The correction $2 \log \delta$ results in the log-likelihood value of the piecewise-constant continuous distribution corresponding to the discrete one and makes likelihood values comparable for different step sizes δ , as well as comparable to the Gaussian log-likelihood.

Method	Feat.	Prop.	T. Obj.	S0		S1		S2	
				L^2 (ln perplexity)		L^2 (ln perplexity)		L^2 (ln perplexity)	
				20	40	20	40	20	40
Linear	-	-	-	12.62	49.07	11.81	42.58	11.86	42.37
Quadratic	-	-	-	6.67	37.95	8.21	46.34	8.35	47.29
<i>SimNet</i>	(V)	-	-	1.21	1.89	1.93	5.16	5.67	24.69
<i>MechaNet1</i>	(V)	LSTM	L2	0.31	25.37	1.52	30.84	-	-
<i>MechaNet2</i>	(T)	conv	L2	0.77	10.68	1.91	34.46	1.95	13.24
<i>MechaNet3</i>	(T)	conv	GL	0.55	15.74	2.26	26.04	-	-
				(0.97)	(385)	(4.08)	(36)		
<i>MechaNet4</i>	(T)	conv	SM	0.56	24.59	2.13	19.54	3.62	9.55
				(0.49)	(0.96)	(3.19)	(17.2)	(5.24)	(11.47)

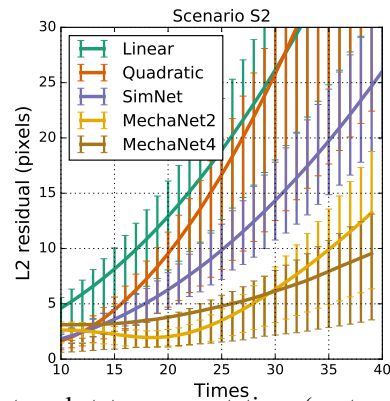


Table 2: **Long term predictions.** (V) and (T) refers to the dimensionality of the internal state representations (vector and tensor respectively). We expect (T) to maintain a 2D spatial model which leads to higher accuracy. The *SimNet* and all *MechaNet* models observed the $T_0 = 4$ first frames as input. All networks have been trained to predict the $T = 20$ first positions, except in Scenario S2, where *MechaNet1* and *MechaNet4* have been trained to predict $T = 30$ frames in order to experience enough variation in the underlying physical conditions, *i.e.*, changing friction. Perplexity (\log_e values shown in the table) is defined as $2^{-\mathbb{E}[\log_2(p(x))]}$ where p is the estimated posterior distribution. *Right*: error evolution on experiment S2 for all time steps up to 40. Error bars denote 25th and 75th percentiles of the L^2 loss in pixels.

Rendering and physical simulation use Blender 2.77’s OpenGL renderer and the Blender Game Engine (relying on Bullet 2 as physics engine) respectively. The object is a cube of side 0.1^3 Blender units with mass = 1. The simulation parameters are: max physics steps = 5, physics substeps = 3, max logic steps = 5, FPS = 120. Rendering used white environment lighting (energy = 0.7) and no other light sources. The object color was set to Lambertian red (RGB: 0.8, 0.04, 0.04) with no specular component. The slope is completely black, covering the whole field of view. The output images were stored as 128×128 color JPG files. See Fig. 1 for an example initial setting from Scenario S2.

4.2. Baseline predictors

Least squares fit. We compare the performance of our methods to two simple least squares baselines: Linear and Quadratic. In both cases we fit two least squares polynomials to the *estimated* screen-space coordinates of the first $T = 10$ frames. The polynomials are of first and second degree(s), respectively. We estimate the object’s position in this case by using the maximum location of the red channel of the input image. Note, that being able to observe the first 10 frames is a large advantage compared to the networks, which only see the first $T_0 = 4$ frames.

Physics simulator. The *SimNet* baseline is used to evaluate the long term prediction ability of a neural network that has access to an explicit physics simulator, in a manner analogous to the work of (Wu et al., 2015). Similarly to the other networks, *SimNet* observes the first T_0 images and aims to regress the physical parameters necessary to predict the trajectory of the object using the physics engine.

The simulator is assumed to have access to a perfect model of the underlying physical laws. The regression architecture constitutes of the vector based feature extraction network described in Section 3.2 with an extra fully-connected layer on top to regress physical parameters. The network is trained with an L^2 loss to infer the current slope rotation angles and friction coefficient $(\theta_x, \theta_y, \rho)$, the object’s position at the observed frames $(t_0, \dots, T_0 - 1)$, and its final velocity at frame $T_0 - 1$.

We input the regressed parameters to the *same* physics simulation system that generated the dataset (Section 4.1) and run the simulation to predict the following object positions $T_0 \dots T$. Note that, since the simulator used by the network is the same as the one used to generate the data, this network is given a significant advantage over the other models.

	MN4	MN4	MN4	MN4	QD	SN
	10	20	30	40	-	-
10	1.18	1.60	0.83	1.00	1.45	1.26
20	11.79	2.13	1.36	1.38	8.21	1.93
30	28.04	6.91	2.71	2.32	23.33	3.23
40	48.65	19.54	8.96	4.00	46.34	5.16

Table 3: **Generalization capabilities.** We compare predictions obtained at different times in Scenario S1 from four versions of the *MechaNet4* (MN4) model that have been trained to predict the first $t = 10, 20, 30,$ and 40 frames (rows). The train and test inputs always consists of the first $T_0 = 4$ images. For comparison, we also show the prediction accuracy of the quadratic baseline (QD) (fit to the first 10 inputs) and *SimNet* (SN). The numbers represent the average, L^2 loss measured in pixels (input image size: 128×128).

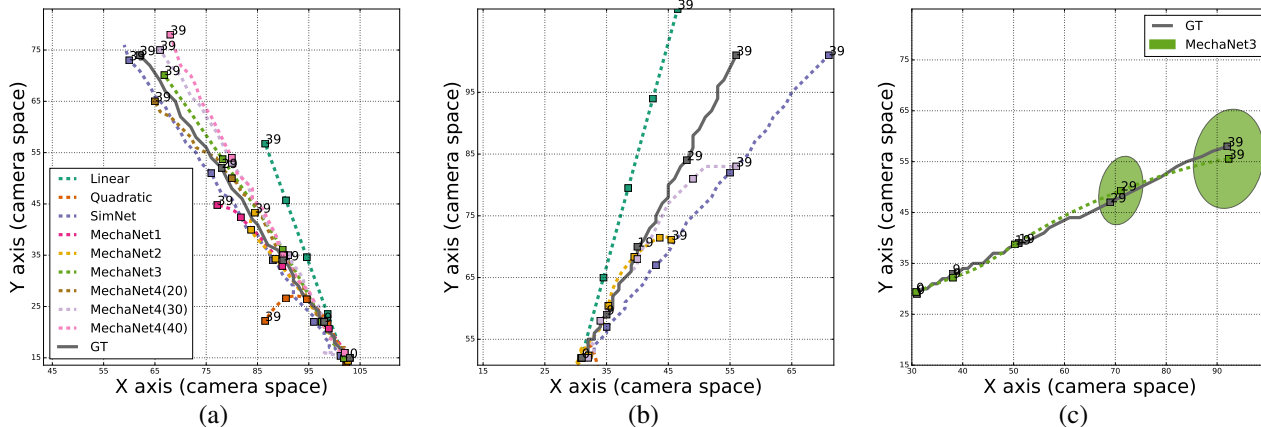


Figure 3: **Qualitative results.** (a) Predictions of various networks in an example drawn from Scenario S1. The number of frames used to train each network is indicated in parenthesis; if omitted, it defaults to 20. (b) The same but for Scenario S2 and focusing on linear and quadratic fits, *SimNet*, *MechaNet2* and *MechaNet4* (30). (c) Example of uncertainty prediction of *MechaNet3* in Scenario S1.

4.3. MechaNets

Experiments consider four different variants of the mechanical prediction networks (*MechaNet1* to 4 for short). *MechaNet1* and *MechaNet2* are trained to optimise the \mathcal{L} ; *MechaNet1* uses the LSTM propagation network and the spatially concentrated internal representation, whereas *MechaNet2* uses the simpler convolutional propagation network but the distributed representation. *MechaNet3* and *MechaNet4* are similar to *MechaNet2*, but they use probabilist predictors, using the Gaussian and probability map outputs, respectively. The four variants are summarized in Table 2.

Implementation details. Network weights are initialized by sampling from a Gaussian distribution. Training uses a batch size of 50 using the first 10 to 40 frames of each video sequence using RMSProp (Tieleman & Hinton, 2012). Training is halted when there is no improvement of the L^2 loss after 40 consecutive epochs; 1,000 epochs were found to be usually sufficient for convergence.

Since during the initial phases of training the network is very uncertain, the model using the Gaussian log-likelihood loss \mathcal{L}_{nm} was found to get stuck on solutions with very high variance $\Sigma(t)$; to solve this issue, the regularizer $\lambda \sum_t \det \Sigma(t)$ was added to the loss, setting $\lambda = 10$ for the first few epochs and then lowering it to $\lambda = 0$ when the value of the determinant stabilized under 100 on average (this variance is comparable to the image size).

In all our experiments we used Tensorflow (Abadi et al., 2015) r0.12 on a single NVIDIA Titan X GPU.

4.4. Results

Long term predictions. Table 2 and Fig. 3 compare the baseline predictors and the four MechaNets on the task of long term prediction of the object trajectory. We call this “long term” in the sense that all methods observe only the first $T_0 = 4$ frames of a video (except the linear and quadratic extrapolators which observe the first 10 frames instead), to then extrapolate the trajectory to 40 time steps.

All networks can be used to perform arbitrary long predictions; the table, in particular, reports the the average L^2 prediction errors at time $T_{\text{test}} = 20$ and 40. However, models in this table are only shown the first $T_{\text{train}} = 20$ frames of each video during training.

Consider first the prediction results for $T_{\text{test}} = T_{\text{train}} = 20$. All networks perform considerably better than the linear and quadratic extrapolators in all scenarios, with error rates 5-40 \times smaller. As expected, Scenario S1 and S2 are harder than Scenario S0, which uses a fixed slope and homogeneous friction, but the network prediction errors are still small, in the order of 1-2 pixels. All networks perform similarly well, particularly in Scenarios S1, with a slight advantage for the LSTM-based propagation networks. *SimNet* is very competitive, as may be expected given that it uses the ground-truth physics engine for integration. However, in Scenario S2 this method does not work well since the variable friction distribution is not observable from the first $T_0 = 4$ frames of a video; *MechaNet2* and *MechaNet4*, which can better learn such effects, can account for such uncertainty and significantly outperform *SimNet*.

Results are different for predictions at time $T_{\text{test}} = 40 \gg T_{\text{train}}$. All networks still outperform the extrapolators, but

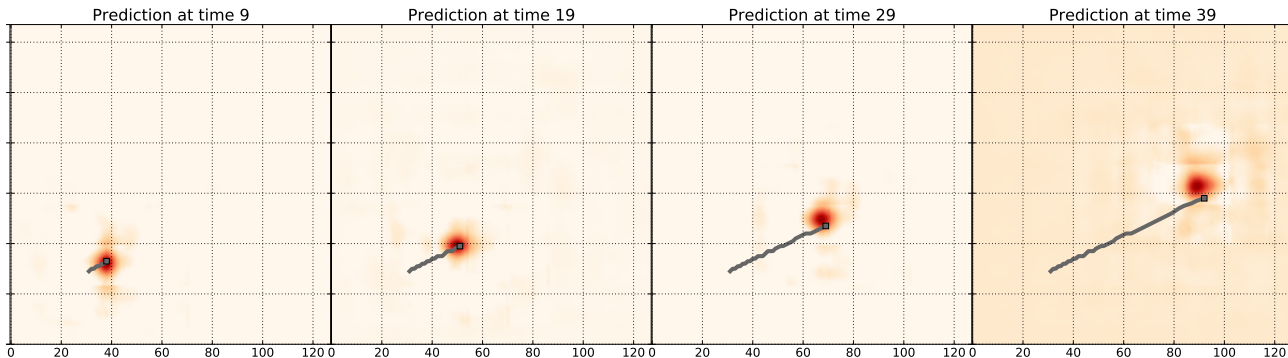


Figure 4: **Uncertainty prediction using probability maps.** The figure shows the output of *MechaNet4* (40) on one example sequence in Scenario S1.

in Scenarios S0 and S1 *SimNet* performs better than the other networks: by having access to the physics simulator, generalization is not an issue. On the other hand, this experiment shows that the deep networks have a limited capacity to generalize physics beyond the regimes observed during training. Among such networks, the ones modelling uncertainty (*MechaNet3* and *MechaNet4*) are able to generalize better. Scenario S2 still breaks the assumptions made by *SimNet*, and the other networks outperform it.

Generalization. The issue of generalization is explored in detail in Table 3, focusing on *MechaNet4* that exhibits the best generalization capabilities. The table reports prediction errors at $T = 10, 20, 30, 40$ for networks trained with video sequence of length $T = 10, 20, 30, 40$ respectively. Recall that predictors always observe only the first $T_0 = 4$ frames of each sequence; the only change is to allow the training loss to assess the predictors’ performance on progressively longer videos during training.

As expected, training on longer sequences dramatically improves the accuracy of longer term predictions, but also the shorter term ones. Training on the full sequences, in particular, performs $\sim 20\%$ better than *SimNet*. This confirms that, while deep networks are able to learn physical rules accurately for the range of physical experiences observed during training, they do not necessarily learn rules that generalize as readily as conventional physical laws.

Predicting uncertainty. *MechaNet3* and *MechaNet4* predict a posterior distribution of possible object locations, using a Gaussian and a probability map model respectively. Table 2 shows that the latter model has significantly lower perplexity, suggesting that the Gaussian model is somewhat too constrained. Qualitatively, Fig. 3 and 4 show that both models make very reasonable predictions of uncertainty, with the uncertain area growing over time as expected.

5. Conclusions

In this paper we explored the possibility of using a single neural network for long-term prediction of mechanical phenomena. We considered in particular the problem of predicting the long-term motion of a cuboid sliding down a slope of unknown inclination and heterogeneous friction. Differently from many other approaches, we use the network *not* to predict some physical quantities to be integrated by a simulator, but to directly predict the complete trajectory of the object end-to-end.

Our results, obtained from extensive synthetic simulation, indicate that deep neural networks can successfully predict long-term trajectories without requiring explicit modeling of the underlying physics. They can also reliably estimate a distribution over such predictions to account for uncertainty in the data. Remarkably, these models are competitive with alternative predictors that have access to the ground-truth physical simulator, and outperform them when some of the physical parameters are not observable or known *a-priori*. However, neural networks exhibit a limited capability to perform predictions outside the physical regimes observed during training. In other words, the internal representation of physics learned by such model is not as general as standard physical laws.

Several future directions remain to be explored. Given the accuracy of mechanical simulators, synthetic experiments are sufficient to assess the capability of networks to learn mechanical phenomena. However, the obvious next phase will be to test the framework on video footage obtained from real-world data in order to assess the ability to do so from visual data affected by real nuisance factors. The other important generalization is to consider more complex physical phenomena, including multiple sliding objects with possible interactions, rolling motion, and sliding over non-flat surfaces.

References

- Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- Agrawal, Pulkit, Nair, Ashvin V, Abbeel, Pieter, Malik, Jitendra, and Levine, Sergey. Learning to Poke by Poking: Experiential Learning of Intuitive Physics. In *Proc. NIPS*, pp. 5074–5082, 2016.
- Battaglia, Peter, Pascanu, Razvan, Lai, Matthew, Rezende, Danilo Jimenez, et al. Interaction networks for learning about objects, relations and physics. In *Proc. NIPS*, pp. 4502–4510, 2016.
- Battaglia, Peter W, Hamrick, Jessica B, and Tenenbaum, Joshua B. Simulation as an engine of physical scene understanding. *PNAS*, 110(45):18327–18332, 2013.
- Chang, Michael B, Ullman, Tomer, Torralba, Antonio, and Tenenbaum, Joshua B. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.
- Cho, Kyunghyun, van Merriënboer, Bart, Gülçehre, Çalar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. EMNLP*, pp. 1724–1734, 2014.
- De Brabandere, Bert, Jia, Xu, Tuytelaars, Tinne, and Van Gool, Luc. Dynamic filter networks. In *Proc. NIPS*, 2016.
- Denil, Misha, Agrawal, Pulkit, Kulkarni, Tejas D, Erez, Tom, Battaglia, Peter, and de Freitas, Nando. Learning to perform physics experiments via deep reinforcement learning. *Deep Reinforcement Learning Workshop, NIPS*, 2016.
- Fragkiadaki, Katerina, Agrawal, Pulkit, Levine, Sergey, and Malik, Jitendra. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015.
- Hamrick, J. B., Battaglia, P. W., Griffiths, T. L., and Tenenbaum, J. B. Inferring mass in complex scenes by mental simulation. *Cognition*, 157:61–76, 2016.
- He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *IEEE CVPR*, pp. 770–778, 2016.
- Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.
- Jeong, SoHyeon, Solenthaler, Barbara, Pollefeys, Marc, Gross, Markus, et al. Data-driven fluid simulations using regression forests. *ACM Trans. on Graphics (TOG)*, 34(6):199, 2015.
- Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, pp. 1097–1105, 2012.
- Lerer, Adam, Gross, Sam, and Fergus, Rob. Learning physical intuition of block towers by example. *arXiv preprint arXiv:1603.01312*, 2016.
- Li, Wenbin, Leonardis, Aleš, and Fritz, Mario. Visual stability prediction and its application to manipulation. *arXiv preprint arXiv:1609.04861*, 2016.
- Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In *IEEE CVPR*, June 2015.
- Monszpart, Aron, Thuerey, Nils, and Mitra, Niloy. SMASH: Physics-guided Reconstruction of Collisions from Videos. *ACM Trans. on Graphics (TOG)*, 2016.
- Mottaghi, Roozbeh, Bagherinezhad, Hessam, Rastegari, Mohammad, and Farhadi, Ali. Newtonian scene understanding: Unfolding the dynamics of objects in static images. In *IEEE CVPR*, 2016.
- Ondruska, Peter and Posner, Ingmar. Deep tracking: Seeing beyond seeing using recurrent neural networks. In *Proc. AAAI*, 2016.
- Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.
- Stewart, Russell and Ermon, Stefano. Label-free supervision of neural networks with physics and domain knowledge. *CoRR*, abs/1609.05566, 2016.
- Tieleman, T. and Hinton, G. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.
- Tompson, J., Schlachter, K., Sprechmann, P., and Perlin, K. Accelerating Eulerian Fluid Simulation With Convolutional Networks. *ArXiv e-print arXiv:1607.03597*, 2016.
- Wu, Jiajun, Yildirim, Ilker, Lim, Joseph J, Freeman, Bill, and Tenenbaum, Josh. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Proc. NIPS*, pp. 127–135, 2015.

Wu, Jiajun, Lim, Joseph J, Zhang, Hongyi, Tenenbaum, Joshua B, and Freeman, William T. Physics 101: Learning physical object properties from unlabeled videos. In *Proc. BMVC*, 2016.

Xue, Tianfan, Wu, Jiajun, Bouman, Katherine L, and Freeman, William T. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Proc. NIPS*, 2016.