GitLab repository for the project: `https://gitlab.com/David-/MLCV/tree/master/Deep%20Learning%20for%20Physical%20Processes`

Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge

Athanasios Raptakis
Raptakis@stud.uni-heidelberg.de

David Elsing
david.elsing@t-online.de

Abstract

In this project we study the use of Deep Learning methods for modeling complex phenomena like those occurring in natural physical processes. The large availability of data gathered on physical phenomena gives motivation for the development of a data intensive paradigm, based on Deep Learning in order to model physical phenomena. Despite considerable success in a variety of applications, the machine learning field is not yet ready to handle the level of complexity required by such problems. In this project we use prior scientific knowledge in order to design efficient Deep Learning models. We implemented an example application, namely Sea Surface Temperature Prediction which belongs to a wider class of differential equations describing a large family of physical phenomena. As a guideline we used the methodology described by Bezenac et al. at [1]. Experiments were conducted using high resolution SST data available by the NOAA6 weather satellite and the results were quantitatively and qualitatively evaluated.

1 Introduction

Traditionally, the modelling of physical processes is made by using techniques based on mathematics or physics. A physical process is a phenomenon which can be described by gradual changes through a series of states. According to systems theory, the state of a dynamical process is the minimum information required to fully describe the future development of a physical phenomenon. In general the dynamics of the state are modeled by ODEs or PDEs and solved by numerical solvers based on various "Numerical Integration" or "Finite Element" methods. However, the use of this traditional math-based paradigm fails when dynamical equations are very difficult or imposible to be derived due to the complexity of the underlying physical laws.
Today due to the abundance of large data, captured by different types of sensors like weather satellites, statistical Machine Learning is emerging as a novel paradigm, which can be used in order to model very complicated systems like Weather, chemical processes etc. Despite impressive success in domains such as vision, language, speech, etc. the Machine Learning approach is not yet ready to challenge the physical paradigm for modeling complex natural phenomena. In their recent work Bezenac et al. [1] proposed a novel Machine Learning technique to model complex physical phenomena. The novelty of their work is the combination of a Convolutional-Deconvolutional Neural Network with prior knowledge gained from physics in order to create an efficient Deep Learning model. During this Project we studied the Bezenac et al. approach at [1] and reproduced their results for Sea Surface Temperature forecasting (SST).

2 Use of Prior Scientific Knowledge

Sea Surface Temperature forecasting (SST) belongs to a more general class of transport problems, the fluid transport problem. In fluids transport occurs through the combination of two principles: advection and diffusion. During advection, a fluid transports some conserved quantity I (the temperature for SST) or material via bulk motion. Diffusion corresponds to the movement which spreads out the quantity I from areas of high concentration to areas of low concentration. The following equation describes the transport of quantity I through advection and diffusion:

$$\frac{\partial I}{\partial t} + (w.\nabla)I = D\nabla^2 I \tag{1}$$

where w denotes the motion vector $\frac{\Delta x}{\Delta t}$, ∇^2 denotes the Laplacian operator and D the diffusion coefficient. This equation describes a large family of physical processes (e.g. fluid dynamics, heat conduction, wind dynamics, etc).The solution of equation 3 is:

Theorem 1 *For any initial condition $I_0 \in L^1(R^2) with I_0(\pm\infty) = 0$, there exists a unique global solution $I(x,t)$ to the advection-diffusion equation, where*

$$I(x,t) = \int_{R^2} k(x-w,y)I_0(y)dy \tag{2}$$

and $k(u,v) = \frac{1}{4\pi Dt}e^{-\frac{1}{4Dt}\|u-v\|^2}$ is a Gaussian probability density with mean $x-w$ and variance $2Dt$.

Quantity $I(x,t)$ can be computed from the initial condition I_0 via a simple convolution with a Gaussian probability density function. Unfortunately neither the initial conditions, the motion vectors nor the diffusion coefficient are known. Inspired from the general solution, Bezenac et al.[1] propose a Deep Learning architecture for predicting SST. The model learns to predict a motion field w, which is used to predict future images.

2.1 Warping Scheme

Discretizing the solution of the advection-diffusion equation by replacing the integral with a sum, and setting image I_t as the initial condition, we obtain a method to calculate the future image, based on the motion field estimate $\hat{w}$. The latter is used as a warping scheme:

$$\hat{I}_{t+1}(x,t) = \sum_{y\in\Omega} k(x-\hat{w}(x),y)I_t(y) \tag{3}$$

where $k(x-\hat{w},y) = \frac{1}{4\pi D\Delta t}e^{-\frac{1}{4D\Delta t}\|x-\hat{w}-y\|^2}$ is a radial basis function kernel, as in equation 4, parameterized by the diffusion coefficient D and the time step value Δt between t and $t+1$.

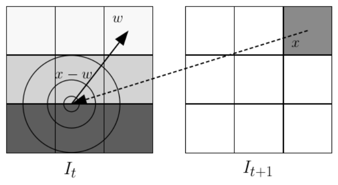

Figure 1: Warping scheme. To calculate the pixel value for time $t+1$ at position x, we first compute its previous position at time t, i.e. $x-w$. We then center a Gaussian in that position in order to obtain a weight value for each pixel in I_t based on its distance with $x-w$, and compute a weighted average of the pixel values of I_t . This weighted average will correspond to the new pixel value at x in I_{t+1}.

The proposed warping mechanism by Bezenac et al.[1] is adapted to the modeling of general advection-diffusion phenomena. Additionally, this warping scheme is entirely differentiable, allowing backpropagation of the error signal to the motion estimating module.

3 Motion Estimation and Forecasting

A convolutional-deconvolutional Neural Network is used to predict a motion vector for each pixel. The network uses skip connections [He et al., 2015], allowing fine grained information from the first layers to flow through in a more direct manner. We use a batch normalization layer between each convolution, and Leaky ReLU (with parameter value set to 0.1) non-linearities between convolutions and transposed-convolutions.

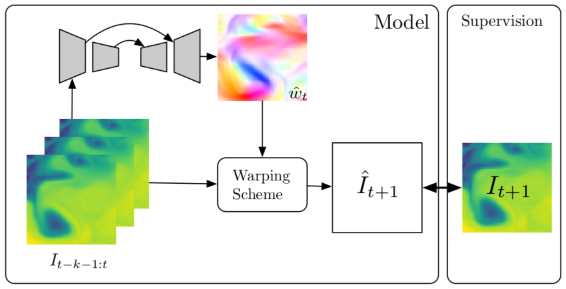

Figure 2: Motion is estimated from the input images with a convolutional neural network. A warping scheme then displaces the last input image along this motion estimate to produce the future image. The error signal is calculated using the target future image, and is backprogated through the warping scheme to correct the CNN. To produce multiple time-step forecasts, the predicted image is fed back in the CNN in an autoregressive manner.

At first the motion field $\hat{w}$ is predicted from a sequence of past input images by a convolutional-deconvolutional (CDNN) module. Next, the last input image is warped using the motion field from the first component, in order to produce the forecast. The entire system is trained in an end-to-end fashion, using only the supervision from the target SST image. Generally, in our problem, we do not have a direct supervision on the motion vector field since the target motion is usually not available. Using the warping scheme we are able to (weakly) supervise $\hat{w}$, based on discrepancy of the warped version of I_t image and the target image I_{t+1}. In order to train our system an input of four ($k = 4$) concatenated images was used.

3.1 Loss function

For our experiment we used Charbonnier function with regularisation terms as proposed at [1]. Charbonnier penalty function $\varrho(x) = (x + \epsilon)^{\alpha}$, evaluates the discrepacy between the warped image $\hat{I}_{t+1}$ and the target image I_{t+1}, where ϵ and α are parameters to be set. Note that with $\epsilon = 0$ and $\alpha = 2$, we recover the $l2$ loss. The Charbonnier penalty function is known to reduce the influence of outliers compared to an $l2$ norm. In order to incorporate prior knowledge to the model one can add penalty terms in the loss function. In our experiments, we used: divergence $\nabla.w_t(x)^2$, magnitude $\|w_t(x)\|^2$ and smoothness $\|\nabla w_t(x)\|^2$. The objective function can be written as:

$$L_t = \sum_{x\in\Omega} \varrho(\hat{I}_{t+1}(x) - I_{t+1}(x)) + \lambda_{div}(\nabla.\hat{w}_t(x))^2 + \lambda_{magn}\|\hat{w}_t(x)\|^2 + \lambda_{grad}\|\nabla \hat{w}_t(x)\|^2 \quad (4)$$

4 Experiments

4.1 CPU Experiment

4.1.1 CDNN architecture

In this section we will describe the CDNN created for the experiment. The network follows the general architecture of a Unet [3] with 4 Convolutional Layers and 4 Deconvolutional Layers with skip connections allowing fine grained information from the first layers to flow through in a direct manner [2]. The output of a "deeper" layer is first upsampled and then concatenated with the skip connections in order to create the input of the deconvolutional layer exactly like a Unet at [3]. Also we have modified the Layers to be residual layers like the ones proposed by He et al. at [2]. The input of our model is k concatenated images $I_{t-k-1:t}$ with (k=4) and the output is the motion field $\hat{w} \in R^{2\times W\times H}$ where W and H is the width and height of the image respectively. We use a batch normalization layer between each convolution, and Leaky ReLU (with parameter value set to 0.1) non-linearities between convolutions and transposed-convolutions. The final layer of this CDNN is only a 2d-convolutional layer with a 3x3 kernel. At the end of every encoder a Max pooling layer was used. We Also tried Average Pooling but there was not any significant difference in performance.

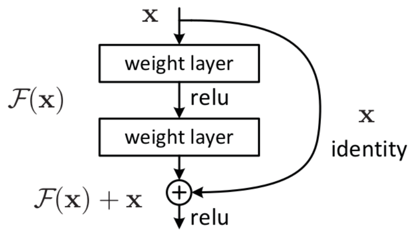

Figure 3: The general form of a residual layer. According to He et al. it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping

4.1.2 Quantitative Evaluation

In order to train our model we used the first 80 percent of the SST data in region 18 (2006 - today) by the NOAA6 weather satellite [4] available by Copernicus at [5]. The rest 20 percent of the data was used for testing. For the training we used MSE loss and the proposed Charbonnier Loss function with $\alpha = \frac{1}{2}, \epsilon = 0, \lambda_{div} = 1, \lambda_{magn} = -0.1, \lambda_{grad} = 0.4$, and Diffusion coefficient $D = 0.45$. We trained the model in a computer with CPU of 3.5GHz and 32Gb of RAM. We did not have a GPU available and thats why we trained only in a tiny part of the available dataset. We tried several batch sizes and we had best performance for batchsize=100 for the MSE loss and batchsize = 12 for the Charbonnier loss. The optimiser we used was Adam with a $learningrate = 10^{-3}$.

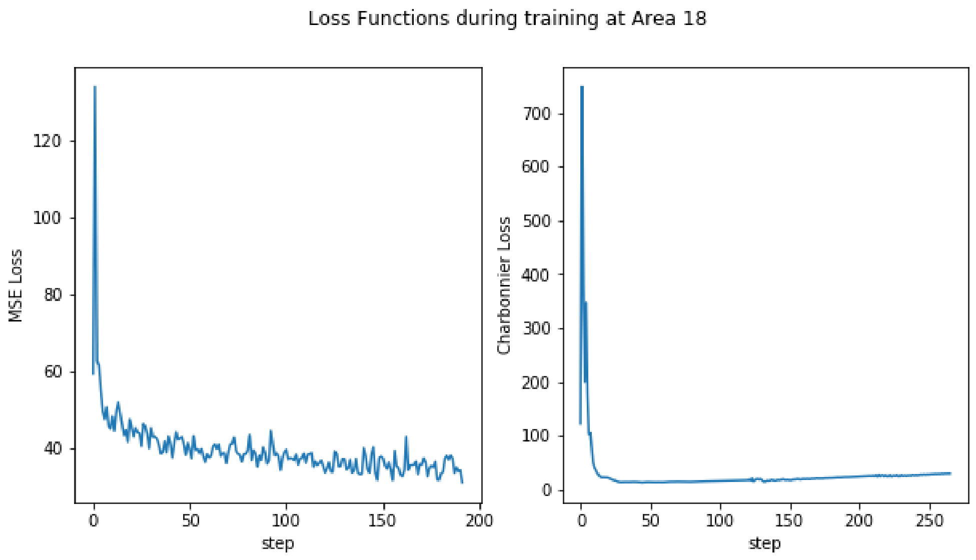

Figure 4: MSE and Charbonnier Loss during training for sevaral epochs >20

To evaluate our model we computed the mean square error (MSE) in a horizon of 1.

4.1.3 Qualitative Evaluation

In order to evaluate the quallity of our model we feed the resulting prediction in the input in an autoregressive manner in order to forecast with a horizon of 6. As you can see in the example bellow,

Table 1: mean square error (MSE) in a horizon of 1.

Loss Function	Average Score (MSE)
MSE	0.012
Charbonnier	0.106

our model trully captures the dynamics of the system. However, because we did not train it with enough data (only 80 percent of area 18) the quality substantially degrades after $\hat{I}_{t+3}$.

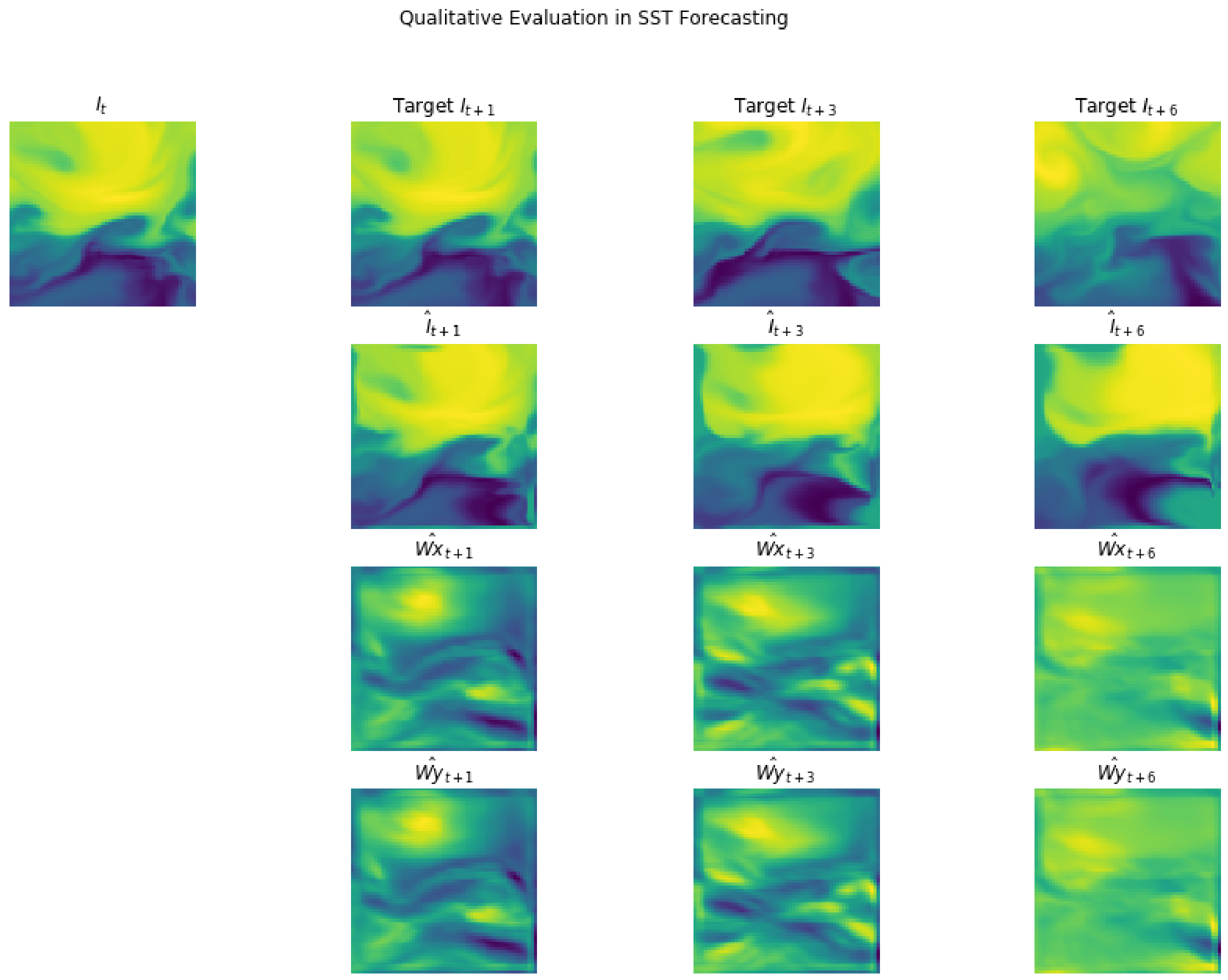

Figure 5: Example of Forecasting in a horizon of 6.From top to bottom:target, our prediction, our model flow $\hat{w}_x, hatw_y$.

4.2 GPU Experiment

During this experiment we used a GPU in order to train our model and optimise the parameters.

4.2.1 Charbonnier loss coefficients

The (average) MSE loss of a small subset of the validation set was evaluated after 10 epochs of training on a small subset of the training set for different values of ϵ and α. The tested values for

ϵ were -1,-0.5,0,0.1,0.5,1 and to; and for α $\frac{1}{6}$, $\frac{1}{5}$, $\frac{1}{4}$, $\frac{1}{3}$, $\frac{1}{2}$and 1. The loss varied between 0.626 for $\epsilon = 0.5$ and $\alpha = \frac{1}{4}$ and 0.924, so theses values were chosen.

4.2.2 Diffusion coefficient

The time step for the solution of the convection-diffusion equation was set to 1, while the best diffusion constant D was determined by setting the velocity w to 0 and testing the sum of the average MSE loss of the minibatches for different values on the validation set:

Comparison between Different Diffusion Coefficients

D	MSE loss
0.01	55211.9
0.1	48.6
1	57.6
2	72.4
10	152.1
0.1	47.2
0.11	42.8
0.12	40.9
0.13	40.1
0.14	39.9
0.15	40.0
0.16	40.2
0.17	40.4
0.18	40.7
0.19	40.9
0.2	41.2

Therefore, $D = 0.14$ was chosen as optimal.

4.2.3 Regulation

We tried using the regulation parameters mentioned in the paper, $\lambda_{\text{div}} = 1$, $\lambda_{\text{grad}} = 0.4$ and $\lambda_{\text{magn}} = -0.03$, but after enough training, the loss became negative, also for different Charbonnier coefficients because of λ_{magn}, after which the results became unusable. Thus, we only trained the net for a longer time without regularisation terms.

4.2.4 Training for many Epochs without regulation

We trained the net with the training dataset and the parameters mentioned above for 36 epochs. The total training Charbonnier loss went down to 31151552 and the average validation MSE loss to 0.053. The predicted image seems to be nearer to the target than to the input image.

4.2.5 Example of a prediction with GPU trained model

It is easy to observe that the model was successfully trained. However, the predicted motion flow $\hat{w}$ is not smooth. A possible remedy to this could be the use of a deeper net.

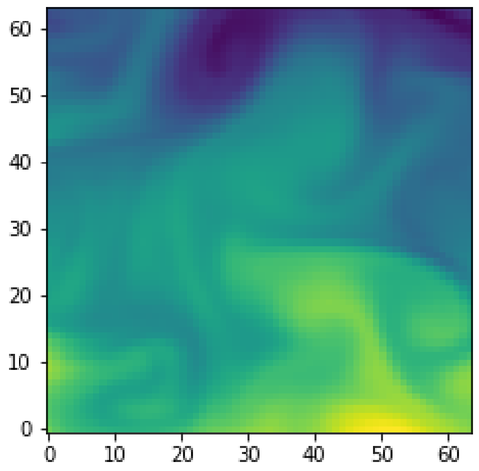

Figure 6: Input $\hat{I}_t$

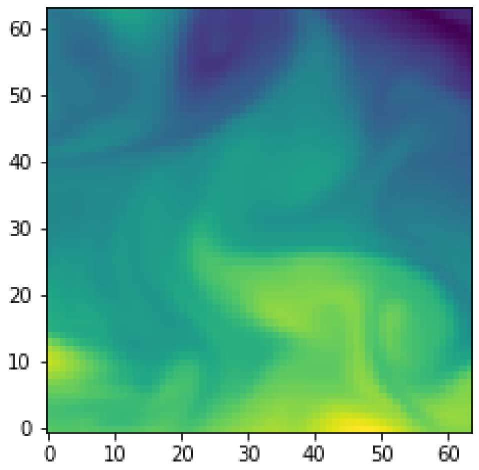

Figure 7: Target Image

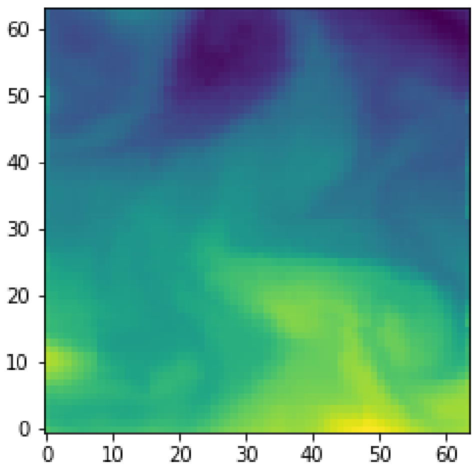

Figure 8: Predicted $\hat{I}_{t+1}$

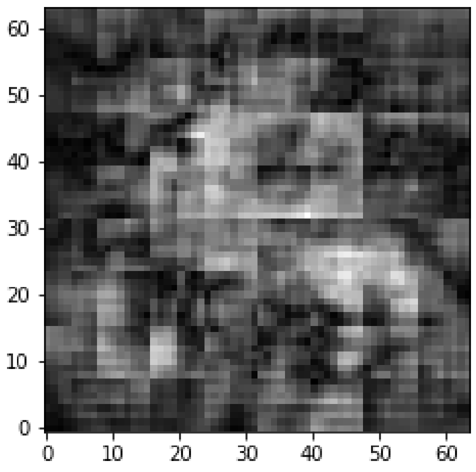

Figure 9: magnitude of flow field $\hat{w}_t$

5 Conclusion

Machine Learning offers a new powerfull alternative to the classic math or physics based approach in the modelling of complex natural processes. During this project we have proved that the combination of prior scientific knowledge and Deep Learning can take advantage of the abundant data available in order to push further the frontier of complex data modelling. The methods we successfully studied and implemented for the example application of SST can be easily generalised in a large category of advection - diffusion processes.

References

[1] Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge
Emmanuel de Bezenac, Arthur Pajot, Patrick Gallinari URL https://arxiv.org/abs/1711.07970.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[3] U-Net: Convolutional Networks for Biomedical Image Segmentation
Olaf Ronneberger, Philipp Fischer, Thomas Brox URL https://arxiv.org/abs/1505.04597.

[4] R. L. Bernstein. Sea surface temperature estimation using the noaa 6 satellite advanced very high resolution radiometer. Journal of Geophysical Research: Oceans, 87(C12):9455–9465, 1982. ISSN 2156-2202. doi: 10.1029/JC087iC12p09455. URL http://dx.doi.org/10.1029/ JC087iC12p09455.

[5] Copernicus Marine environment monitoring service
URL http://marine.copernicus.eu/