Latent-space Physics: Towards Learning the Temporal Evolution of Fluid Flow

Steffen Wiewel, Moritz Becher, Nils Thuerey
Technical University of Munich

Motivation

- The field of computational methods has been highly successful at developing powerful **numerical algorithms** to predict how the natural phenomena under consideration will behave.

- Take a different view:
  - instead of relying on analytic expressions
  - use a deep learning approach to infer physical functions based on data
  - focus on temporal evolution of complex functions in the context of fluid flows

Challenge: high dimensionality of Eulerian space-time data sets

Contributions

- a first LSTM architecture to predict temporal evolutions of dense, 3D functions of physics system in latent spaces
- a neural-network(an encoder and decoder architecture) based simulation algorithm with significant practical speed-ups. (a strong compression)
    - more than two orders of magnitudes faster than a traditional pressure solver
- a detailed evaluation of training modalities
- Give NNs predictive capabilities for complex inverse problems

Background: flow physics

- *Navier-Stokes* (NS) model

$$\partial \mathbf{u}/\partial t + \mathbf{u}\cdot\nabla\mathbf{u} = -1/\rho\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{g}\,, \quad \nabla\cdot\mathbf{u} = 0, \qquad (1)$$

u: flow velocity

p: pressure

ρ, ν, **g**: density, kinematic viscosity and external forces

Method

- **Goal**: predict future states of a physical function **x** ($R^3 \times R^+ \rightarrow R^d$):
  - d = 1 for scalar functions such as pressure
  - d = 3 for vectorial functions such velocity fields
  - Can represent a spatio-temporal pressure function, or a velocity field

- **Key**: reducing the dimensionality of the problem using *convolutional neural networks* (CNNs) with respect to both **time and space**

- Process:
  - Map the original 3D problem into a smaller spatial *latent space;* Learn the inverse mapping at the same time
  - Maps a collection of reduced representations into an encoding of the temporal evolution
  - Reduced temporal state is used to output a sequence of spatial latent space representations
  - Decode latent space representations to yield the full spatial data set

Method (cont'd)

- Goal: Given a functional representation f_t , a current state and *n* previous states, predict future states from $\mathbf{x}(t + h)$ to $\mathbf{x}(t + oh)$: $$f_t\left(\mathbf{x}(t-nh),...,\mathbf{x}(t-h),\mathbf{x}(t)\right) \approx \left[\mathbf{x}(t+h),\ldots,\mathbf{x}(t+oh)\right]. \quad (2)$$

- Employ f_d (CNN / decoder) and f_e (CNN / encoder) to compute low dimensional encodings Rephrase Eq. (2):
$$\begin{aligned}\tilde{f}_t\left(f_e(\mathbf{x}(t-nh)),...,f_e(\mathbf{x}(t))\right) &\approx \left[\mathbf{c}^{t+h},\ldots,\mathbf{c}^{t+oh}\right] \\ f_d\left(\left[\mathbf{c}^{t+h},\ldots,\mathbf{c}^{t+oh}\right]\right) &\approx \left[\mathbf{x}(t+h),\ldots,\mathbf{x}(t+oh)\right] \quad (3) \\ f_d\left(\tilde{f}_t\left(f_e(\mathbf{x}(t-nh)),...,f_e(\mathbf{x}(t))\right)\right) &\approx f_t(\mathbf{x}(t-nh),...,\mathbf{x}(t))\end{aligned}$$

- **c**: latent space, $\mathbf{c} \in \mathbb{R}^{m_s}$

- $\tilde{f}_t$: the prediction network, also an encoder-decoder structure
  - turns the temporal stack of encoded data points $f_e(\mathbf{x})$ into a reduced representation **d**, refer to as *temporal context.*

Architecture (spatial network)

Goal: Reducing Dimensionality

- a fully convolutional autoencoder

$$\min_{\theta_d,\theta_e} |f_d(f_e(\mathbf{x}(t))) - \mathbf{x}(t)|_2, \quad (4)$$

θ_d,θ_e: parameters of the decoder and encoder

- Note that: paths from f_{e_k} over c_k to f_{e_5} are only active in pretraining stage k. After pretraining only the path f_{d_k} over c_5 to f_{d_5} remains active.

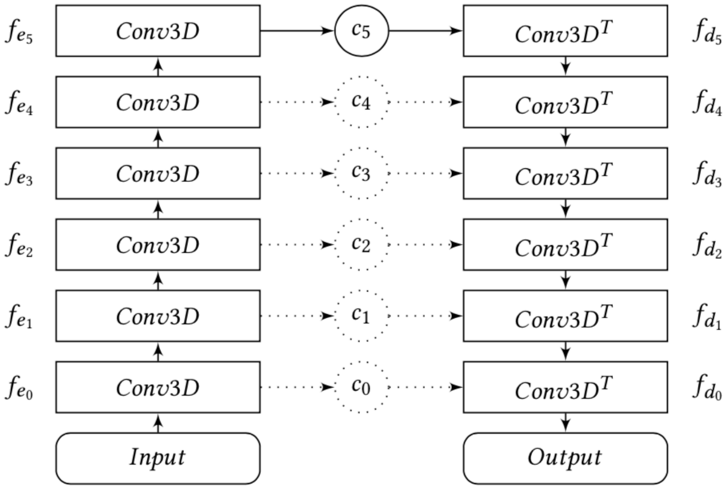


Fig 1: Overview of the autoencoder network

Architecture (temporal network)

Goal: Prediction of Future States

- LSTM layers for predicting the evolution over time.
  - Minimize the mean absolute error between the *o* predicted and ground truth states:

$$\min_{\theta_t} \left| \tilde{f}_t \left(\mathbf{c}^{t-nh}, \ldots, \mathbf{c}^{t-h}, \mathbf{c}^{t} \right) - \left[\mathbf{c}^{t+h}, \ldots, \mathbf{c}^{t+oh} \right] \right|_1 . \qquad (6)$$

- a recurrent encoder module (LSTM)
  - transform n + 1 latent space points into a temporal context **d**
- a recurrent decoder module with similar structure (layers of LSTM units)
  - takes a context d as input, outputs a series of future, spatial latent space representations

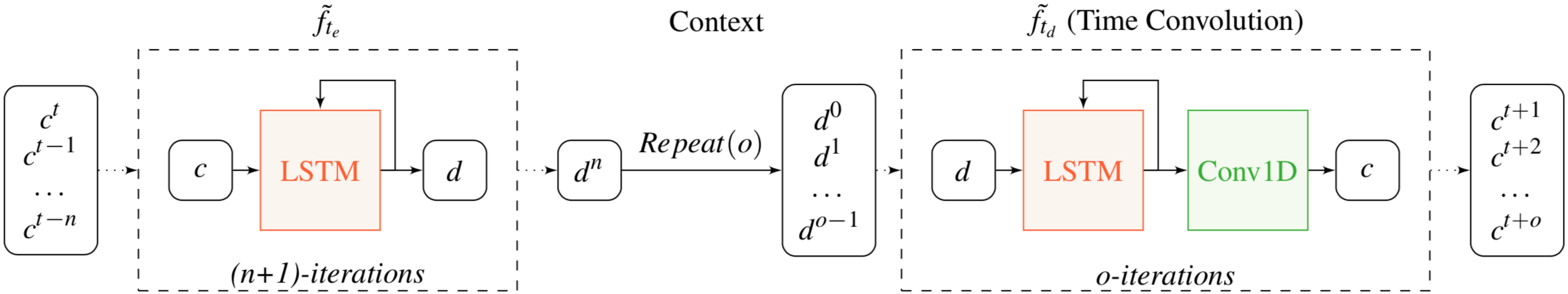
✓ During encoding, only keep the very last context, pass it to the decoder part.

✓ During decoding, context is repeated *o* times for the decoder LSTM to infer the desired future states.

Architecture (temporal network)

Goal: Prediction of Future States

- To prevent overfitting: a hybrid structure of LSTM units and convolutions



dashed boxes indicate an iterative evaluation

Navier-Stokes (NS) model:

$$\partial \mathbf{u}/\partial t + \mathbf{u}\cdot\nabla\mathbf{u} = -1/\rho\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{g}\,,\quad \nabla\cdot\mathbf{u} = 0,$$

Datasets: Fluid Flow Data

- Rely on a NS solver with operator splitting to calculate the training data at discrete points in space and time
  - Computing motion of the fluid with the help of transport, i.e. *advection* steps, for the velocity **u**, evaluating external forces **g**, then computing the harmonic pressure function p.
  - A visible, passive quantity such as smoke density ρ, or a level-set representation φ for free surface flows is advected in parallel to the velocity itself.

  - Consider a *split* pressure. a hydrostatic pressure p_s for cell at height z
  - a hydrostatic pressure p_s for cell at height z:

$$p_s(z) = p(z_0) + \frac{1}{A}\int_z^{z_0}\iint_A \mathbf{g}\rho(h)\ \mathrm{d}x\mathrm{d}y\ \mathrm{d}h$$

    with z_0, p_0, A denote surface height, surface pressure, and cell area.
  - Density and gravity can be treated as constant in this setting, simplifies to: $p_s = \rho\mathbf{g}(z - z_0)$
    - Can be evaluated efficiently
    - Drawback: only valid for fluids in hydrostatic equilibrium, typically cannot be used for dynamic simulations in 3D

Datasets: Fluid Flow Data (cont'd)

- Given a data-driven method to predict pressure fields, we can incorporate the hydrostatic pressure into a 3D liquid simulation:
  - decomposing the regular pressure field p_t into: $p_t = p_s + p_d$
  - p_s: hydrostatic component
  - p_d: dynamic component
  - our autoencoder separately receives and encodes the two fields p_s and p_d.

- With split pressure, the autoencoder could potentially put more emphasis on the small-scale fluctuations p_d from the hydrostatic pressure gradient.

Datasets:

- Three 3D data sets:

	liquid64	liquid128	smoke128
Scenes n_s	4000	800	800
Time steps n_t	100	100	100
Size	419.43GB	671.09GB	671.09GB
Size, encoded	1.64GB	2.62GB	2.62GB

- use randomized simulation setups, target scenes with high complexity:
  - strong visible splashes and vortices,
  - large CFL (Courant- Friedrichs-Lewy) numbers (how fast information travels from cell to cell in a complex simulation domain)

- For each data sets:
  1. generate n_s scenes of different initial conditions
  2. discard the first n_w time steps (typically contain small, regular, and less representative dynamics)
  3. store a fixed number of n_t time steps: a final size of $n_s n_t$ spatial data sets.
  4. Normalize each dataset to [-1,1].

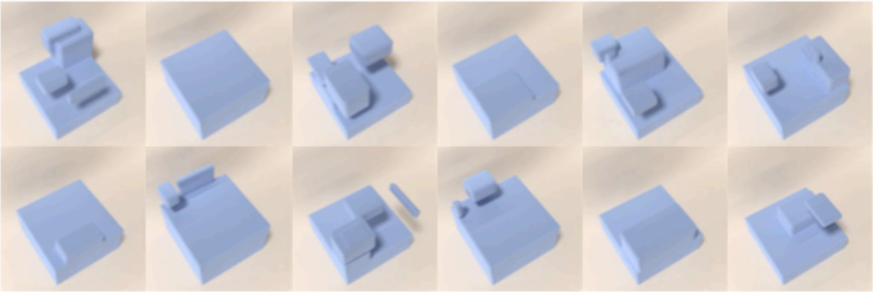


Figure 17: *Examples of initial scene states in the* liquid64 *data set.*

	liquid64	liquid128	smoke128
Scenes n_s	4000	800	800
Time steps n_t	100	100	100
Size	419.43GB	671.09GB	671.09GB
Size, encoded	1.64GB	2.62GB	2.62GB

Figure 18: *Examples of initial scene states in the* liquid128 *data set. The more complex initial shapes are visible in several of these configurations.*

Figure 19: *Examples states of the* smoke128 *training data set at* $t = 65$.

Interval Prediction

- Use network prediction for a time interval of i_p time steps
- Then perform a single full simulation step without any network calculations
    - numerical time integration + network prediction
    - Our prediction intervals i_p on the order of 4 to 14 steps
    - $i_p = 0$: the LSTM prediction is not used at all
    - ip = ∞: the fully recurrent LSTM

Evaluation and Training

- PSNR (peak signal-to-noise ratio) as a baseline metric
- a surface-based Hausdorff distance
- surface error:

$$e_h = \max(1/|S_p| \sum_{\mathbf{p}_1 \in S_p} \phi_r(\mathbf{p}_1), 1/|S_r| \sum_{\mathbf{p}_2 \in S_r} \phi_p(\mathbf{p}_2))/\Delta x.$$

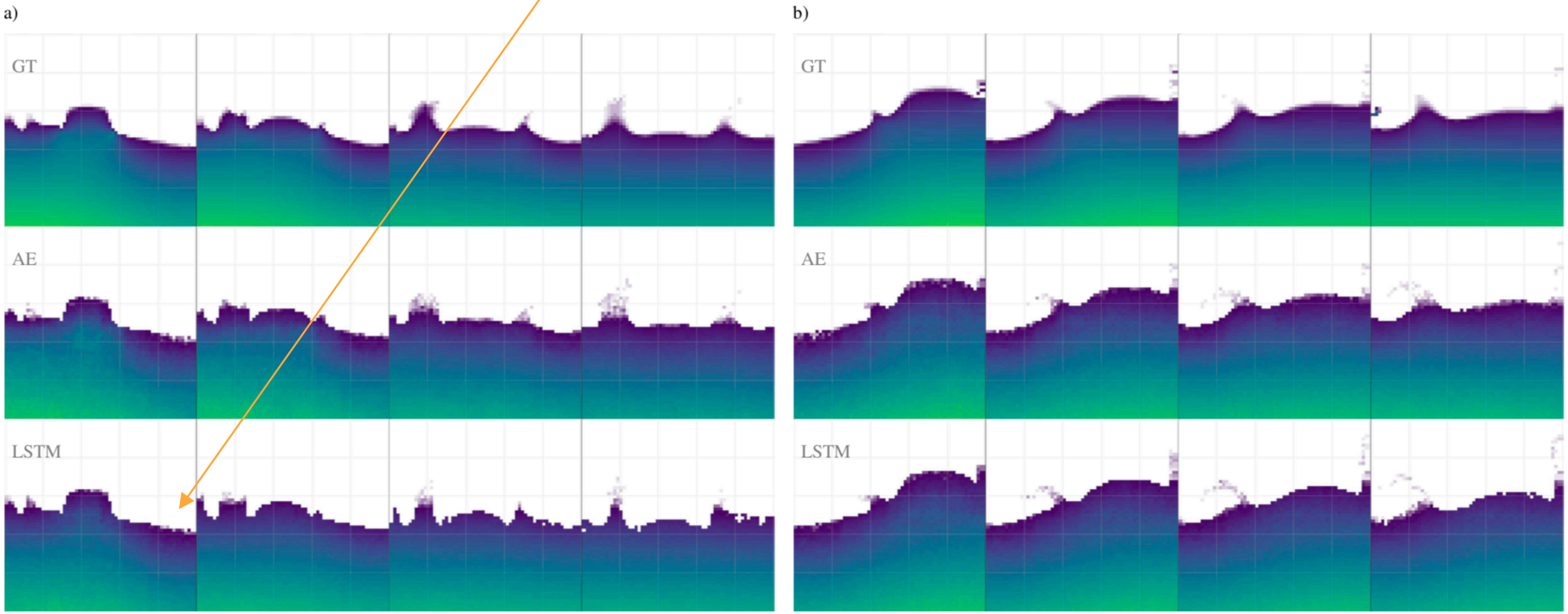
where φ_r, φ_p are two signed distance functions

- Spatial Encoding
  - the autoencoder network in conjunction with a numerical time integration scheme
  - encoder: $\mathbf{c} = f_e(\mathbf{x})$; decoder: $\mathbf{x'} = f_d(\mathbf{c})$
- Temporal Prediction
  - a quantity x' is inferred based on a series of previous latent space points

the network successfully learned an abstraction of the temporal evolution of the flow.

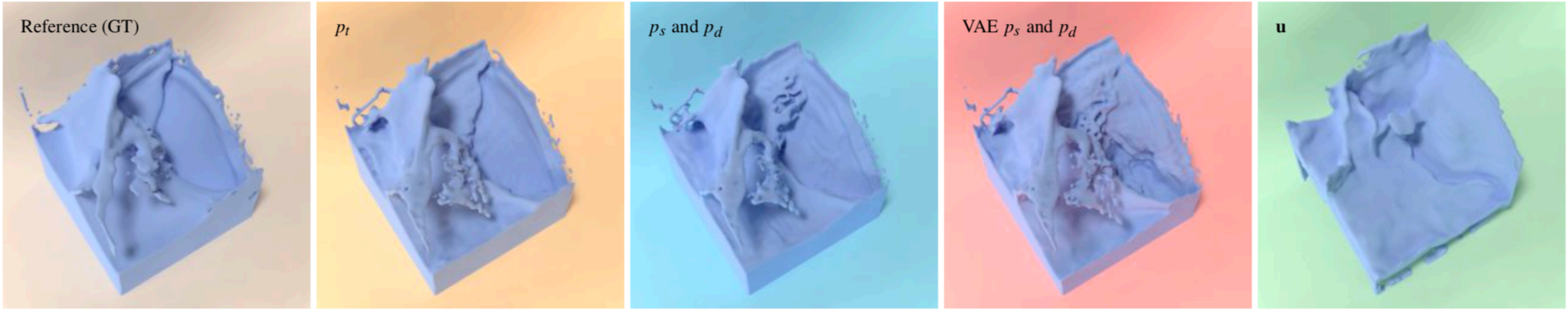
Results

- Compare ground truth, autoencoder baseline and outputs of our prediction network:
  - The temporal predictions closely match the autoencoder baseline.
  - Network can reproduce complex behavior of the underlying simulations.



Results

- Liquid surfaces predicted by different models for 40 steps with $i_p = \infty$.
  - The velocity version (green) leads to large errors in surface position, all three pressure versions closely capture the large-scale motions.
  - On smaller scales, both split pressure variants (p_s and p_d) introduce artifacts



Results

- large speed-ups and robust simulations for a significant variety of fluid scenes

Interval i_p	Solve	Mean surf. dist	Speedup
Reference	2.629s	0.0	1.0
4	0.600s	0.0187	4.4
9	0.335s	0.0300	7.8
14	0.244s	0.0365	10.1
∞	0.047s	0.0479	55.9

	Enc/Dec	Prediction	Speedup
Core exec. time	4.1ms + 3.3ms	9.5ms	155.6

Performance of ten liquid128 scenes, averaged over 150 simulation steps each. The mean surface distance is a measure of deviation from the reference per solve.

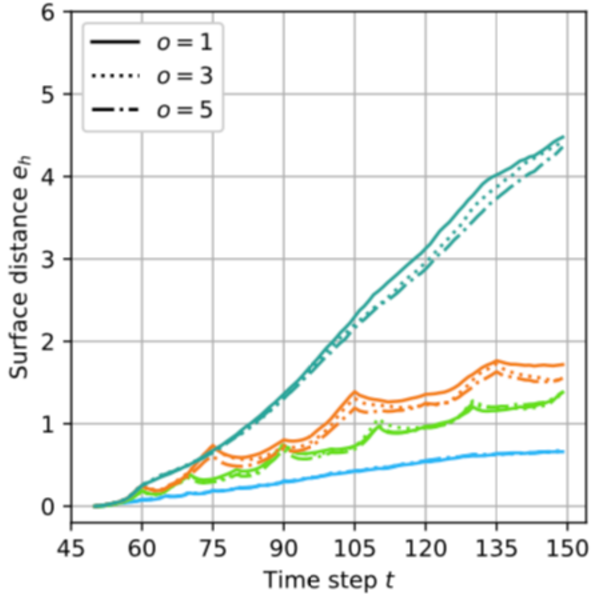
	Solve		Speedup
Reference	169ms		1.0
	Enc/Dec	Prediction	
Core exec., $o = 1$	3.8ms + 3.2ms	9.6ms	10.2
Core exec., $o = 3$	3.9ms + $3 * 3.3$ms	12.5ms	19.3

Performance of ten liquid64 scenes, averaged over 150 simulation steps each.

Results

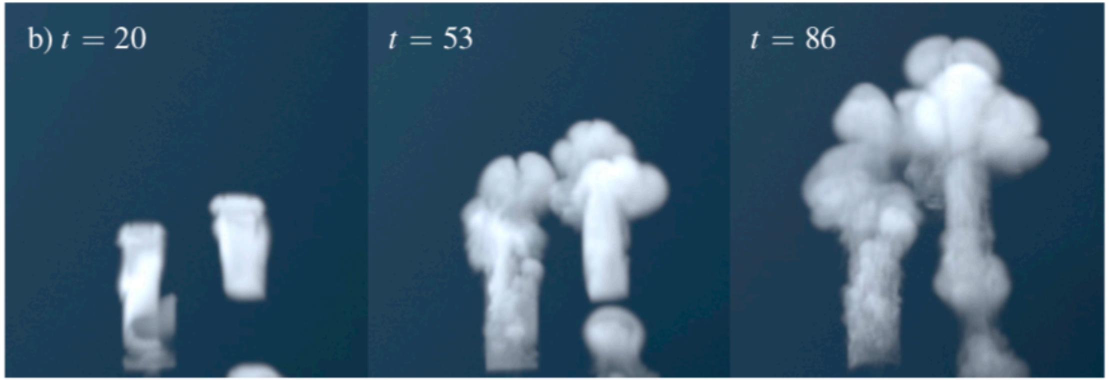
- How well the model can predict future states based on a single set of inputs. For multiple output steps ($o > 1$), our model predicts several latent space points from a single time context **d**. Accuracy for 1, 3 and 5 steps of output:

  - Accuracy barely degrades when multiple steps are predicted at once;
  - But more efficient. $o = 3$ prediction only requires 30% more time to evaluate, despite generating three times as many predictions



Results

- A trained model for the *smoke128* data set.



- Despite the significantly different physics, our approach successfully predicts the evolution and motion of the vortex structures
    - However, we noticed a tendency to underestimate pressure values, and to reduce small-scale motions.

Conclusions

- Limitations
  - LSTM strongly relies on AE (which primarily encodes large scale scale dynamics) while small scale dynamics are integrated by the alignment of free surface boundary conditions
  - Current simple AE can introduce a certain amount of noise; can be alleviated by different network architectures

- Conclusions
  - Neural network architectures can predict the temporal evolution of dense physical functions
  - learned latent spaces with LSTM-CNN hybrids are suitable for this task
  - Significant increases in simulation performance
    - more than 150x faster than a regular pressure solve
    - represents an important first step towards deep-learning powered simulation algorithms