nature machine intelligence

Article

https://doi.org/10.1038/s42256-023-00746-x

Development of the Senseiver for efficient field reconstruction from sparse observations

In the format provided by the authors and unedited

Supplementary Information for The Senseiver: efficient field reconstruction from sparse observations

1 Architecture Details

Attention Block. We use two specific flavors of the multi-head attention mechanism (cross-attention and self-attention) [1] to build the Attention block, which is used in the encoder of the Senseiver. Here, we are outlining the equations for the attention mechanism and detailing the attention block. We use scaled dot product attention,

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\mathbf{V}, \tag{1}$$

where d_k is the dimension (i.e. number of columns) of $\mathbf{K}$ and $\mathbf{Q}$. Multi-head attention applies a set of N_H separate linear transformations to $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ and computes the attention mechanism on each linear transformation h. It then concatenates the outputs from each head and computes a final linear transformation to form the output.

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\mathbf{O}_1 \oplus \cdots \oplus \mathbf{O}_{N_H}]\,\mathbf{W}_M \tag{2}$$

$$\text{where } \mathbf{O}_h = \text{Attention}(\mathbf{Q}\mathbf{W}_h^Q, \mathbf{K}\mathbf{W}_h^K, \mathbf{V}\mathbf{W}_h^V). \tag{3}$$

The attention block, denoted AttentionBlock($\mathbf{X}$; $\mathbf{Q}_{in}, \theta, \phi$) first processes an input, $\mathbf{X}^{(0)} \in \mathbb{R}^{M\times N}$ with a multi-head cross-attention Eq. 2, in which $\mathbf{Q} = \mathbf{Q}_{in} \in \mathbb{R}^{N_{\mathbf{Q}_{in}}\times N}$ and $\mathbf{K} = \mathbf{V} = \mathbf{X}^{(0)} \in \mathbb{R}^{N_{Q_{in}}\times N}$. The output is $\mathbf{X}^{(1)}$ which is processed by an MLP($\mathbf{X}^{(1)}$; θ) with weights θ with three layers to make $\mathbf{X}^{(2)} \in \mathbb{R}^{N_{Q_{in}}\times N}$. The array $\mathbf{X}^{(2)}$ is passed through multi-head self-attention with $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{X}^{(2)}$, and therefore preserves the dimension $N_{Q_{in}} \times N$. Finally, the output of this self-attention is processed by second MLP($\mathbf{X}^{(3)}$; ϕ) with weights ϕ, which preserves the dimensions as $N_{Q_{in}} \times N$. The MLP is composed by two fully connected layers with a GeLU non-linearity in between.

1.1 Design rationale

Key differences between our approach and the Perceiver model is that we wanted to work with a variable number of sensors at inference time, and we aimed only use a small number of parameters. The latter point is particularly critical for applications where computational resources and/or memory are constrained. On the other hand, it is natural to ask why certain design choices were made. For example: Why does the AttentionBlock in equation 6 have its own weights while weights are shared in equations (7-8)? We chose not to share weights in equation (6) to allow for more flexibility in the model and to enable it to learn more complex representations since this is what encodes the input. We chose to share weights for equations 7 and 8 a result of our experimentation. We did explore versions of the model with additional layers, but found that the model with shared weights across two applications of equations (7,8) provided superior results. This suggested to us that this version of the model was more easily trainable. To the former point, the choice to use our specific form of attention, rather than a more canonical cross-attention structure, was influenced by our desire to develop a model capable of dealing with a variable number of sensors, a key feature of our work. While we acknowledge that exhaustive exploration of all possible architectural configurations was beyond our computational resources, our model, as designed, significantly outperforms the state-of-the-art in our tests. As such, we did not see a compelling need to further optimize the hyperparameter beyond the experiments in the next section.

1.2 Computational complexity

Similar to all models that leverage dot-product attention, the computational complexity is primarily determined by the matrix multiplication operations of the queries, keys and values. However, in the encoder of our model, the attention block $\mathbf{E}^{(2)}$ attends to the input only after the input has been compressed through $\mathbf{E}^{(1)}$, and cross attends with a latent array, further circumventing the computational expense of self attention. The encoder has a complexity of $\mathcal{O}(N_s \times N_{Q_{in}} \times N_c)$. N_c is a hyperparameter that compresses the latent array and its impact is studied in Table A1. This circumvents the scaling of $\mathcal{O}(N_s \times N_{Q_{in}} \times (D \times 2 \times N_f + N_I))$ of vanilla cross-attention. Another key innovation of our model is the decoder. Designed to predict one pixel at a time, it ensures a very low complexity of $\mathcal{O}(N_{Q_{in}} \times N_c)$, fostering efficiency and speed, allowing only to predict regions of interest. This circumvents a possible complexity of $\mathcal{O}(N_{output} \times N_s \times N_{Q_{in}} \times (D \times 2 \times N_f))$, where N_{output} would be the number of pixels that one would like to output. This approach has two significant benefits. Firstly, it allows the model to handle extremely large arrays efficiently. This is particularly useful when dealing with multi-channel inputs - for instance, from a sensor gathering multiple types of data simultaneously or very large spatial embeddings. Secondly, it allows the model to seamlessly process a variable number of sensor inputs, which could be key during the

inference stage when the available set of sensors might vary. The compression leads to the significant decrease in the number of parameters of the Senseiver compared to the Perceiver in Table 3 and a much faster inference and training as stated above. To sum it up, the model offers computational efficiency and faster inference times without compromising the benefits of high-resolution spatial definitions. It does this by implementing design choices compressing the input and latent arrays and reusing attention blocks, making it a standout choice for handling large and complex datasets. We have enhanced Perceiver IO [2] to make it more suitable for field-ready devices such as this resulting in a model that requires 1-2 orders of magnitude less parameters with comparable or better performance in these applications.

2 Hyperparameter experiments

The Table 1 shows an extensive hyperparameter study where 5 different sensor configurations are trained with 25 models with different numbers of parameters. On Table 2 we show the impact of the hyperparmater N_f and on Table 3 a comparison with Perceiver IO [3] is shown.

Furthermore, we conducted computational speed experiments on a single thread of a CPU, which is relevant for potential deployment in field devices. In our experiments, for the a 256x128 domain size, the Perceiver (Nf=128) model took 19.4 minutes to train over 50 frames for 10 epochs, with an inference time of 144 minutes on a test set of 5,000 frames. On the other hand, the Senseiver (Nf=128) model trained in 2.7 minutes and had an inference time of 21 minutes for the same dataset and frame count. Additionally, for the a domain size of 100x100x100, the Perceiver model took 528 minutes to train over 50 frames for 1 epoch, while its inference time was 1.1 minutes on a single test frame. Comparatively, the Senseiver model trained in 18 minutes and had an inference time of 0.51 minutes for the same dataset and frame count.

3 Additional experiments

Fig. 1 shows sample reconstructions of the test set with different sensor configurations.

Figure 3 shows the impact in performance and uncertainty with coverage in time (by having more training data) and coverage in space (by having more sensors).

Additionally, we have conducted experiments with two cases of interest (sensors at the wall and sensors located at the upstream). In the upstream sensor placement scenario, 24 sensors were introduced on the second column of the image (y-direction), which resulted in an error of 0.422. This marked a considerable improvement from the error observed with the same number of randomly placed sensors, indicating the benefits of this arrangement. Similarly, in the wall-mounted sensor scenario, we positioned 128 sensors on the first row along the x-axis, yielding an error of 0.295. This performance was on par with tests involving the same number of randomly placed sensors. This suggests that

Table 1: Mean test error (average of 4950 test frames) of models trained using 1% of the available data. The performance for different number of channels per layer (N_c) and size of the latent array (**latents**) is assessed for 5 different sensor configurations. With a very small number of channels (4) the model does not benefit of increasing the latent size. In models with more capacity (more channels) the model's accuracy increases substantially by having a bigger latent array without increasing the parameter count by a large margin. Due to limited computational resources, one model was trained per sensor configuration. On the first column we list the datasets that use the specific Senseiver configuration.

			sensor configuration					
N_c	**latents**	**params**	**16**	**8**	**4**	$\mathbf{8}_{bc}$	$\mathbf{8}_{cyl}$	**dataset**
4	8	2,341	0.69	0.61	0.59	0.68	0.60	
	16	2,373	0.67	0.57	0.60	0.59	0.61	
	32	2,437	0.52	0.57	0.61	0.56	0.61	
	256	3,333	0.57	0.60	0.63	0.63	0.50	
	1024	6,405	0.66	0.50	0.62	0.61	0.61	
8	8	6,441	0.21	0.25	0.29	0.25	0.24	
	16	6,505	0.23	0.24	0.25	0.17	0.29	
	32	6,633	0.15	0.23	0.24	0.18	0.23	
	256	8,425	0.12	0.11	0.14	0.18	0.09	
	1024	14,569	0.08	0.07	0.09	0.08	0.08	
16	8	19,921	0.14	0.12	0.13	0.15	0.12	
	16	20,049	0.10	0.10	0.07	0.08	0.08	
	32	20,305	0.05	0.07	0.06	0.07	0.07	
	256	23,889	0.02	0.02	0.04	0.06	0.03	sea, cylinder
	1024	36,177	0.03	0.02	0.03	0.06	0.02	
24	8	40,441	0.10	0.08	0.10	0.13	0.08	
	16	40,633	0.08	0.06	0.05	0.12	0.06	
	32	41,017	0.05	0.05	0.04	0.09	0.03	
	256	46,393	0.02	0.02	0.03	0.07	0.02	pore
	1024	64,825	0.02	0.01	0.02	0.08	0.01	
32	8	68,001	0.08	0.08	0.06	0.07	0.06	
	16	68,257	0.04	0.05	0.04	0.08	0.06	
	32	68,769	0.04	0.04	0.03	0.06	0.03	
	256	75,937	0.02	0.02	0.02	0.04	0.02	turbulence
	1024	100,513	0.02	0.02	0.02	0.06	0.02	

an organized arrangement of sensors could potentially be more advantageous than random placements, especially when dealing with a restricted number of sensors. Furthermore, we conducted tests on scenarios where the sensors were exposed to noisy readings. We introduced Gaussian noise influenced by a constant κ to the model's inputs as $s_n = s_{n_{GT}} + \kappa n$, where $s_{n_{GT}}$ represents the ground-truth value, and n is the Gaussian noise. Our model significantly outperforms the previous approach [4], with a mean error of 0.232 and 0.367 for kappa values of 0.05 and 1, respectively. This corresponds to an improvement

Table 2: Mean test error and standard deviation (of 4950 test frames) of models trained using 1% of the available data. The performance of different number of spatial components ($\mathbf{N}_f$) is assessed for 5 different sensor configurations. Due to limited computational resources, one model was trained per sensor configuration, hence there is some variability in the results. By adding spatial components the parameter count goes up slightly which results in a a lower mean error in the test set and also a smaller standard deviation in the errors of the predictions.

		sensor configuration				
$\mathbf{N}_f$	**params**	**16**	**8**	**4**	$\mathbf{8}_{bc}$	$\mathbf{8}_{cyl}$
		Mean Error ± Standard Deviation				
32	20,305	0.051 (± 0.042)	0.069 (± 0.094)	0.065 (± 0.066)	0.072 (± 0.077)	0.069 (± 0.095)
64	24,401	0.056 (± 0.039)	0.059 (± 0.085)	0.048 (± 0.044)	0.066 (± 0.020)	0.044 (± 0.075)
128	32,593	0.037 (± 0.031)	0.045 (± 0.031)	0.050 (± 0.033)	0.083 (± 0.015)	0.039 (± 0.060)

Table 3: Mean test error and standard deviation (of 4950 test frames) of models trained using 1% of the available data. The parameter count, computational time and performance of the unmodified Perceiver IO [3] is shown.

			time [minutes]		**sensor configuration**				
Model	$\mathbf{N}_f$	**params**	$t_{10 \text{ epochs}}$	$t_{\text{test set}}$	**16**	**8**	**4**	$\mathbf{8}_{bc}$	$\mathbf{8}_{cyl}$
					Mean Error				
Senseiver	128	32,593	2.7	21	0.037	0.045	0.050	0.083	0.039
PerceiverIO	128	1,188,193	19.4	144	0.048	0.053	0.081	0.097	0.087
					Difference				
		36x	7x	7x	30%	18%	62%	17%	123%

of 20% and 70%, underlining the model's robustness and its potential for effective deployment in practical, real-world situations. Furthermore, we conducted experiments with the PerceiverIO model, obtaining error values of 0.281 and 0.406, respectively.

4 Additional dataset details

4.1 Multiphase flow through porous media

Three snapshots of the dataset are shown in Fig. 4.

4.2 3D Isotropic turbulence

The dataset consists of a 3D Direct Numerical Simulation of homogeneous, isotropic turbulence with passive scalars advected with the flow, in a domain of size 128^3. A passive scalar with an enforced Probability Density Functions is considered here in order to provide more complexity to the case. The complete

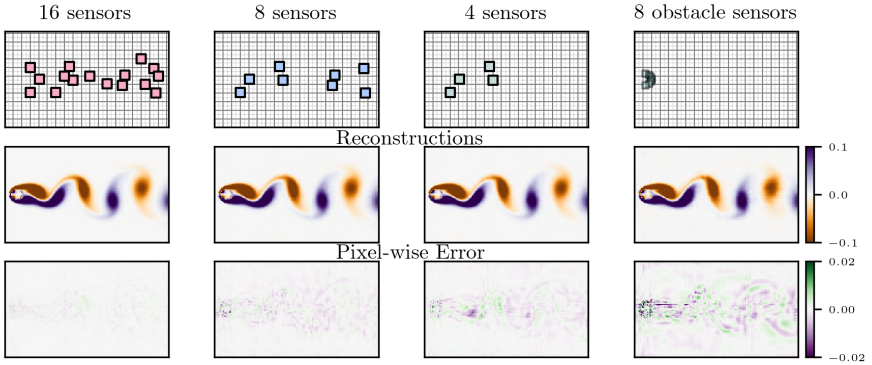

Fig. 1: Reconstruction examples and pixel-wise error of one frame of the test set.

Table 4: Sea temperature test error comparison. We compare the error of the test set obtained with one set of unseen sensors (ϵ) per number of sensors (N_s) of the VoronoiCNN with the average error ($\bar{\epsilon}$) of the Senseiver, averaging over an ensemble of different sensor placements. The extrapolation capabilities of the trained model is shown in degrees Celsius.

	VoronoiCNN	Senseiver					
				Average Degrees Celsius Error			
Ns	ϵ	$\bar{\epsilon}$	**Std**	**2001-03**	**2004-08**	**2009-13**	**2014-18**
10	0.319	0.047 ±	3.07E-03	0.39	0.40	0.42	0.43
20	0.274	0.043 ±	1.78E-03	0.35	0.37	0.39	0.40
30	0.171	0.041 ±	8.17E-04	0.34	0.35	0.37	0.39
40	-	0.040 ±	6.31E-04	0.33	0.34	0.36	0.38
50	0.136	0.040 ±	1.02E-03	0.33	0.34	0.36	0.37
60	-	0.039 ±	6.43E-04	0.33	0.33	0.35	0.37
70	0.112	0.039 ±	4.82E-04	0.33	0.33	0.35	0.37
80	-	0.039 ±	3.11E-04	0.33	0.33	0.35	0.37
90	-	0.039 ±	2.93E-04	0.32	0.33	0.35	0.37
100	0.108	0.039 ±	1.54E-04	0.32	0.32	0.35	0.36

description of the simulation setup is provided in [5]. For simplicity in interpretation, the scalar lower and upper bounds in this dataset are set between -1 and +1 respectively $-1.0 < \phi < 1.0$. A typical example of this type of problem would involve the use of a drone with capabilities to sense methane or a magnetic field in an attempt to locate an undocumented orphan well. There are hundreds of thousands of these wells across the United States and they cause problems by leaking, e.g., methane (a greenhouse gas 80 times worse than carbon dioxide on a 20-year time scale) and hydrogen sulfide (which is deadly even at low concentrations) into the atmosphere. The correlation between velocity magnitude and concentration is shown in Fig. 5.

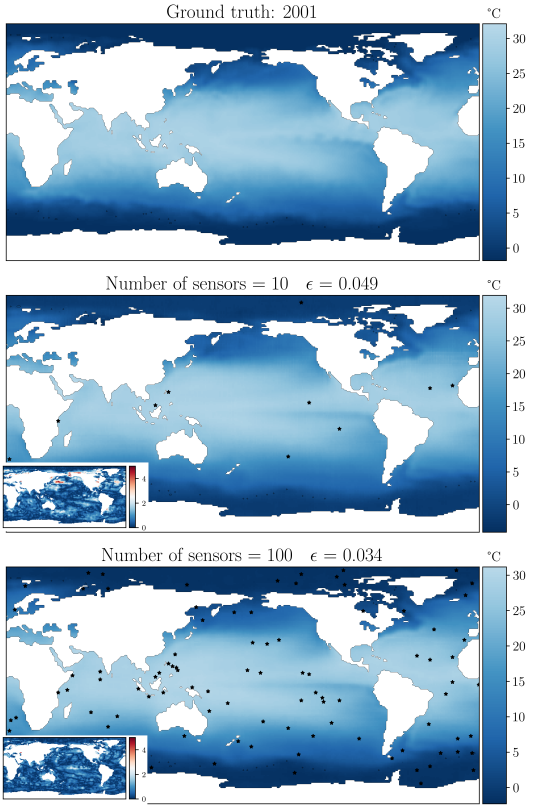

Fig. 2: Results in the NOAA sea temperature dataset. Ground truth of year 2001 (test set) vs reconstruction using ten sensors and one hundred sensors. The bottom left panel shows the difference map between the ground truth in degrees Celsius.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need. Advances in Neural Information Processing Systems **2017-Decem**(Nips), 5999–6009 (2017)

[2] Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., Carreira, J.: Perceiver: General Perception with Iterative Attention (2021)

[3] Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., Hénaff, O., Botvinick, M.M., Zisserman, A., Vinyals, O., Carreira Deepmind, J.: Perceiver IO: A General Architecture for Structured Inputs & Outputs (2021). https://doi.org/10.48550/arxiv.2107.14795

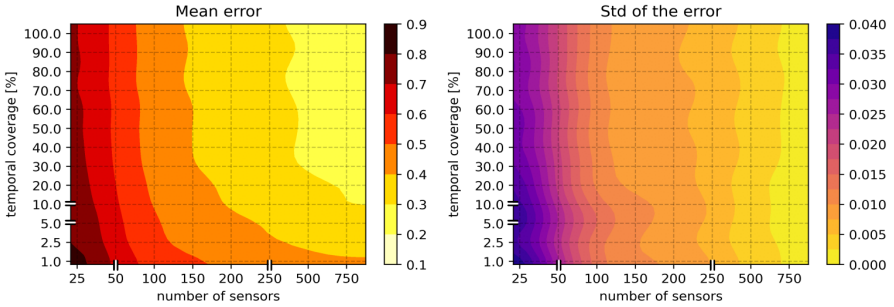

Fig. 3: Impact of number of sensors and temporal coverage during training in the performance of the model in the channel turbulence dataset. **Left**) Mean error throughout the dataset **Right**) Standard deviation of the error in the different time frames. Intuitively, the addition of sensors measurements decreases uncertainty (here shown with the standard deviation) and increases model performance (lower mean error).

[4] Fukami, K., Maulik, R., Ramachandra, N., Fukagata, K., Taira, K.: Global field reconstruction from sparse sensors with Voronoi tessellation-assisted deep learning. Nature Machine Intelligence 2021 3:11 **3**(11), 945–951 (2021). https://doi.org/10.1038/s42256-021-00402-2

[5] Daniel, D., Livescu, D., Ryu, J.: Reaction analogy based forcing for incompressible scalar turbulence. Physical Review Fluids **3**(9), 094602 (2018). https://doi.org/10.1103/PHYSREVFLUIDS.3.094602/FIGURES/9/MEDIUM

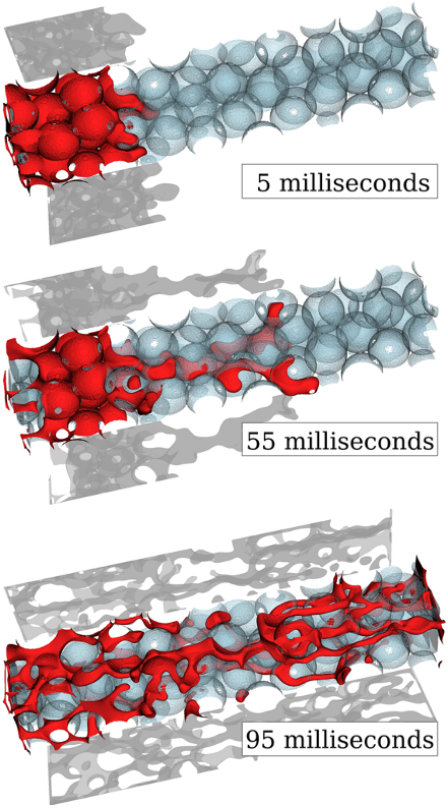

Fig. 4: Three 3D snapshots of the simulation taken at 5, 55 and 95 milliseconds. The solid boundaries are shown with light blue, the invading fluid is shown with red and its shadows with gray. The other fluid, which occupies the complementary space, is not shown.

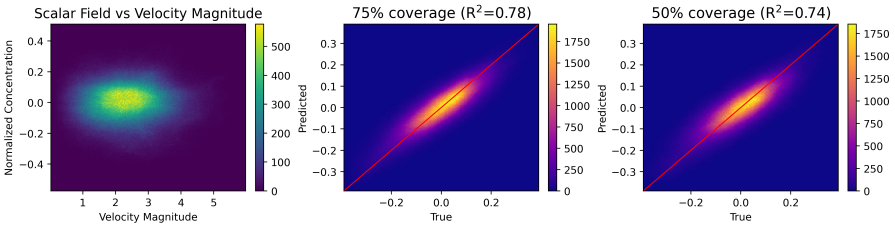

Fig. 5: a) Correlation between velocity magnitude and concentration. The correlation is not trivial. b) Pixel-wise true vs predicted value of our model trained with 75% data of the spatial data c) Pixel-wise true vs predicted value of our model trained with 50% data of the spatial data.