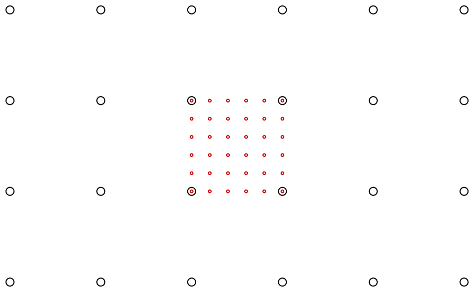
Subgrid Scales and Neuron Nets

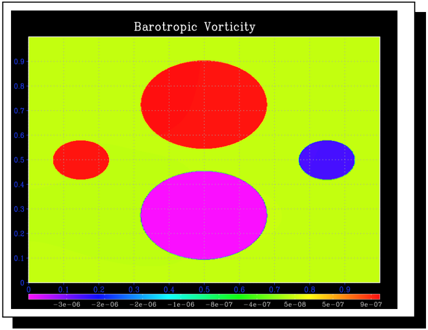
Eugene Kazantsev,
INRIA, AirSea, Grenoble



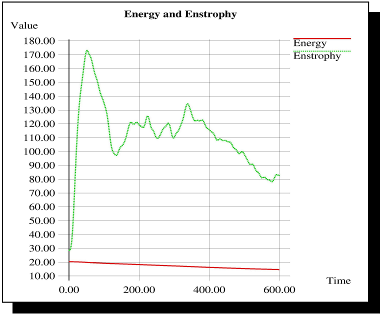
Barotropic equation

$$\frac{\partial \omega(x,y,t)}{\partial t} + J(\psi, \omega + \beta y) = \mu \Delta \omega, \quad \Delta \psi = \omega$$

In a square box $L = 1000\text{km}$, with $\beta = 2 \times 10^{-11} s^{-1}$, $\mu = 10 \frac{m^2}{s}$.
Impermeability and slip boundary conditions ($\psi = \omega \mid_{\text{bnd}} = 0$)
200 days spin-up from 4 different vortices. Resolution 500×500 (2 km)

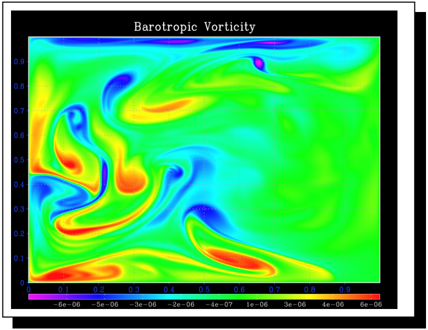


Initial relative vorticity ω

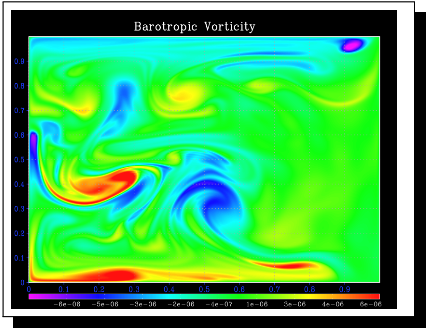


Energy and enstrophy evolution during 600 days

Examples



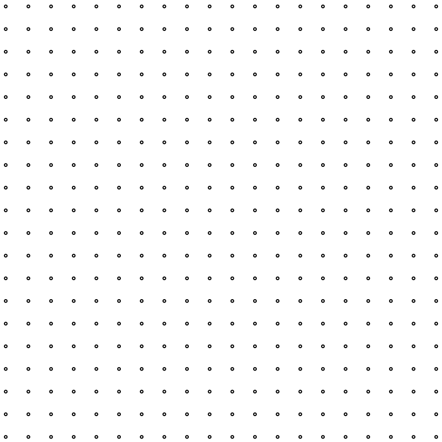
Relative vorticity ω on 403 day



Relative vorticity ω on 428 day

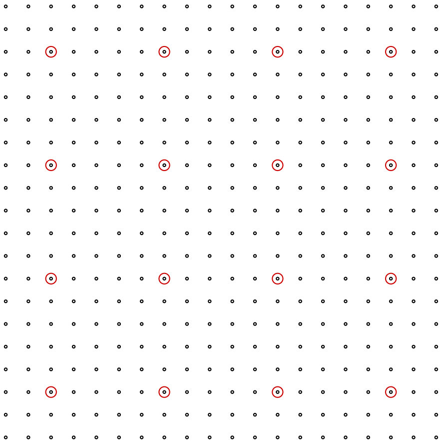
HR and LR Grids

- Consider a 2 km grid and a 10 km grid



HR and LR Grids

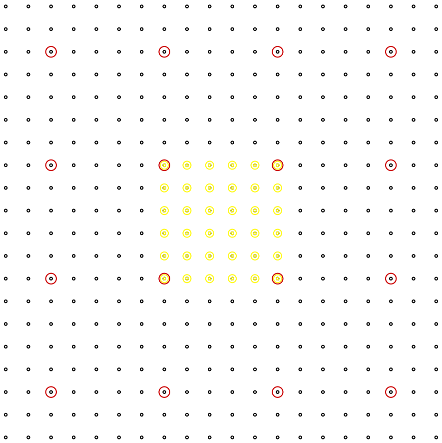
- Consider a 2 km grid and a 10 km grid
- take 16 values on the 10 km grid
-



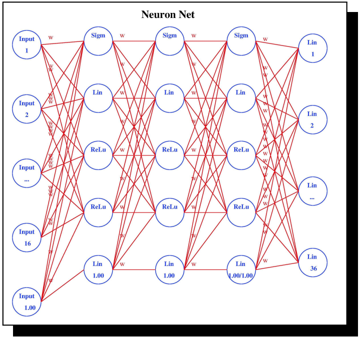
HR and LR Grids

- Consider a 2 km grid and a 10 km grid
- take 16 values on the 10 km grid
- and try to reconstruct 36 values in the central cell by learning model data.

$$Output_{36} = NN(Input_{16})$$



Neuron net



- Fully connected NN
- with 16 inputs + bias and 36 outputs
- Between 1 and 4 hidden layers
- Between 16 and 72 neurons in each hidden layer,
- Sigmoid, Linear, ReLu, Leaky ReLu activations in various combinations.

Data and procedure

Data Sets

- for every x_i from 10 km to 990 km by 10 km (99)
- for every y_j from 10 km to 990 km by 10 km (99)
- for every t_k from 0 to 600 days by 3 hours (4800)

47 000 000 data sets

Procedure

- Learn all data sets:
- Test on 500 000 supplementary sets (t_k from 600 to 660)

$$Err_{NN} = |NN(Input_{16}) - Model_{36}|$$

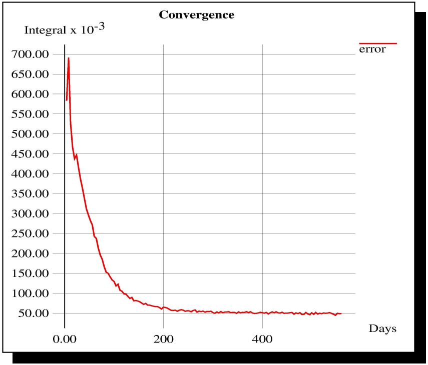
- Compare with Bicubic interpolation

$$Err_{Int} = |Int(Input_{16}) - Model_{36}|$$

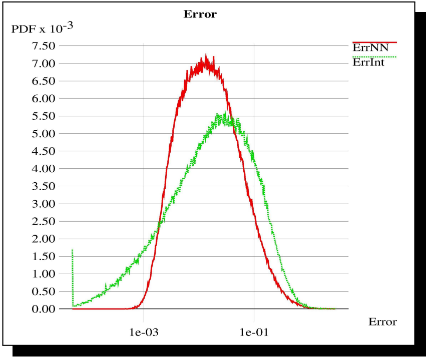
All $Input_{16}$ and $Model_{36}$ are scaled to be in $[-0.5, 0.5]$

$$Input_{16} = 0.5 \frac{Input_{16}}{\max_{16}|Input|}, \quad Model_{36} = 0.5 \frac{Model_{36}}{\max_{16}|Input|}$$

Typical convergence and error distribution.

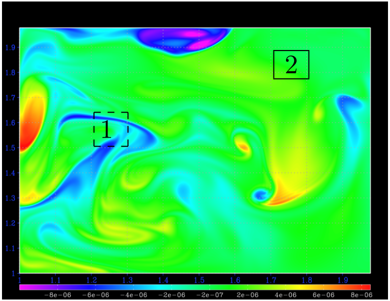


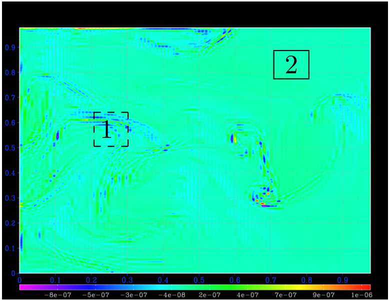
Convergence of Err_{NN} during learning



PDFs of NN error and Interpolation error
Average ErrNN: 0.0342
Average ErrInetrp: 0.0521

$$\text{Model}_{2km} \Longrightarrow \text{Coarse grid}_{10km} \Longrightarrow NN(\text{Model}_{10km})$$

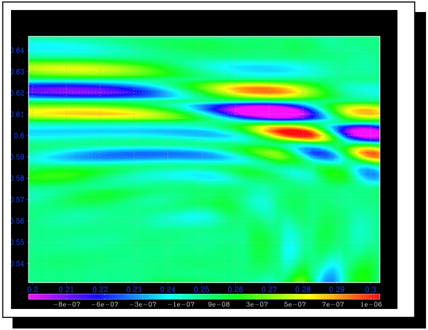




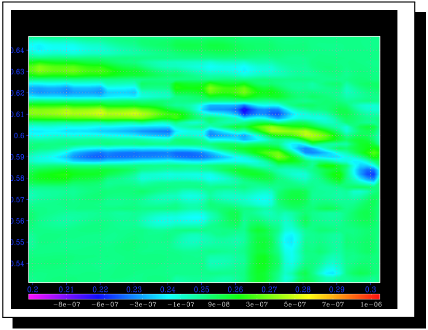
$\omega_{2km}(t = 200 \text{ days})$
From -8×10^{-6} to 8×10^{-6}

$\omega_{2km}(t = 200)$ - NN$(\omega_{10km}(t = 200))$
From -8×10^{-7} to 1×10^{-6}

First Zoom

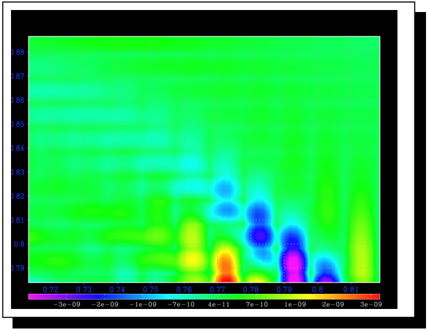


Difference Model—interpolation
From -8×10^{-7} to 8×10^{-7}

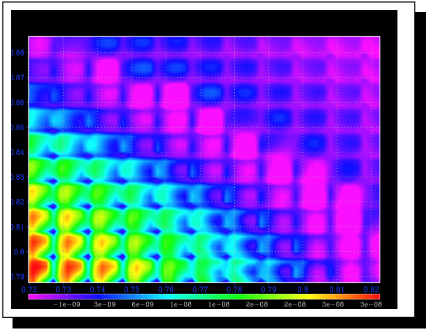


Difference Model—Neuron Net Approx.
From -8×10^{-7} to 8×10^{-7}

Big gradients, big errors, NN is more efficient.



Difference Model—interpolation
From -3×10^{-9} to 3×10^{-9}



Difference Model—Neuron Net Approx.
From -1×10^{-9} to 3×10^{-8}

Small gradients, small errors, a lot of noise in NN.

Conclusions — Questions

We have a Neuron Net that gives a slightly better approximation of the model solution on a fine grid than bicubic interpolation.

- Is it sufficient?
- How to improve?
  - Use NN in high gradient regions and interpolation elsewhere.
  - Use more sophisticated scalar product during learning $< A(NN - Truth), NN - Truth >$ to measure the error norm.
  - Something else?

Used Techniques

- Between 1 and 4 hidden layers,
- Between 16 and 72 neurons in hidden layers,
- With or without bias,
- Original 16 or Interpolated values used as input $Err_{NN} = |NN(Input_{16}) - Model_{36}|$ $Err_{NN} = |NN(Int(Input_{16})) - Model_{36}|$
- Sigmoid, Linear, ReLu, Leaky ReLu activations in various combinations, [a]
- Simple gradient descent or using momentum impact, [b]

[a] R.Prajit *et al*, Searching for Activation Functions, arXiv eprint 1710.05941

[b] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. Neural Networks: The Official Journal of the International Neural Network Society, 12(1), 145–151

- Using Adam method, [a]
- Gradient vanishing compensation[b]
- Enhanced learning for the most difficult cases (more iterations for big errors),
- Neuron dropout [c]
- Multiple use of data sets,
- Working with uniformely scaled data,
- Learning rate decrease. [d]

[a] Diederik P. Kingma and Jimmy Lei Ba. Adam : A method for stochastic optimization. 2014. arXiv:1412.6980v9

[b] R.Pascanu *et al* , On the difficulty of training Recurrent Neural Networks, 2012, arXiv:1211.5063

[c] N. Srivastava,*et al*, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research; 15(56):1929−1958, 2014.

[d] Ruder, Sebastian (2017). "An Overview of Gradient Descent Optimization Algorithms". arXiv:1609.04747