Deep Neural Nets meets O/PDE's

Eldad Haber, UBC

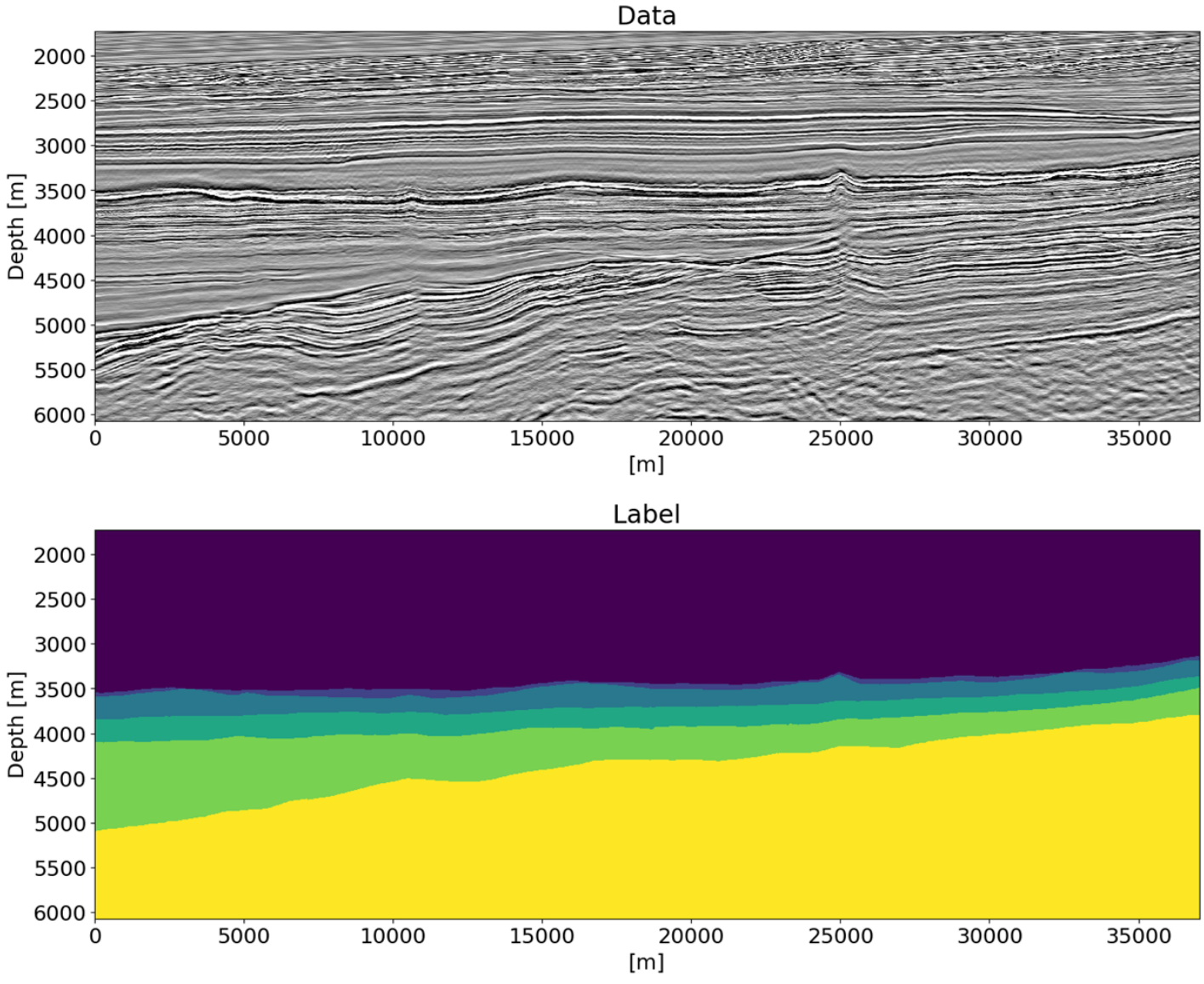
August 21, 2019

- Deep Neural Networks (DNN) and geoscience applications
- Meet Ordinary Differential Equations
- New architectures for DNN's
- Meet Partial Differential Equations
- New architectures for CNN's
- Application
- Future work

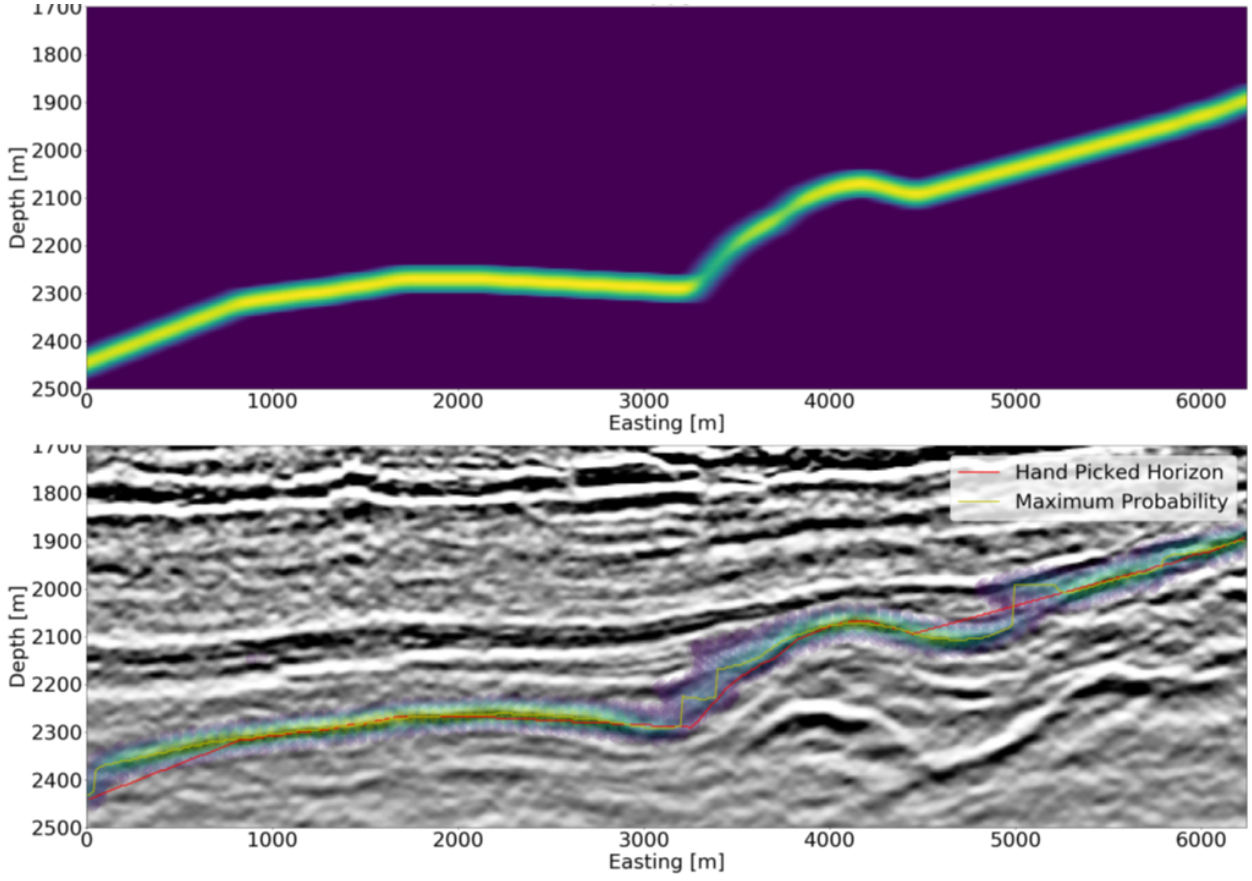
Image Classification

Semantic Segmentation

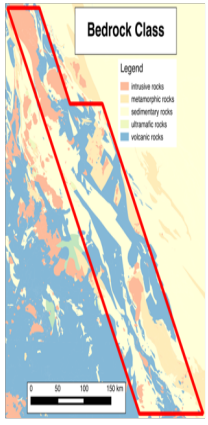
Seismic Semantic Segmentation

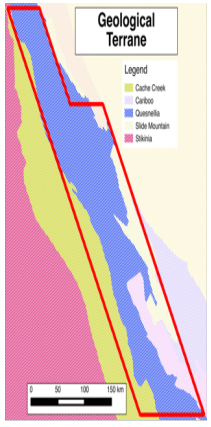


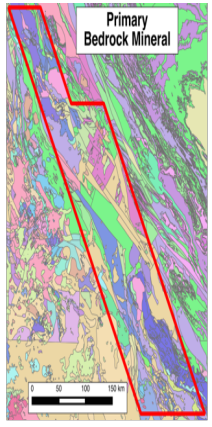
Horizon Detection in Seismic Data

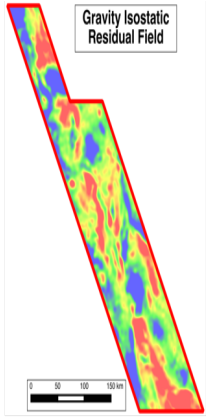


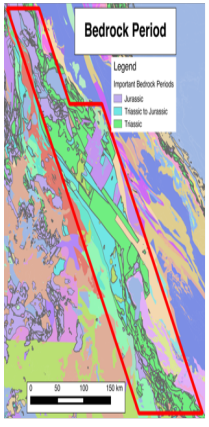
Resource Prospectivity

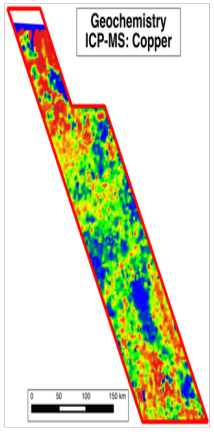


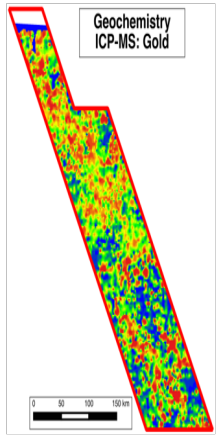


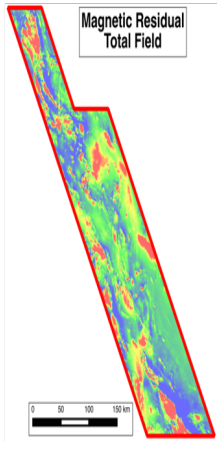


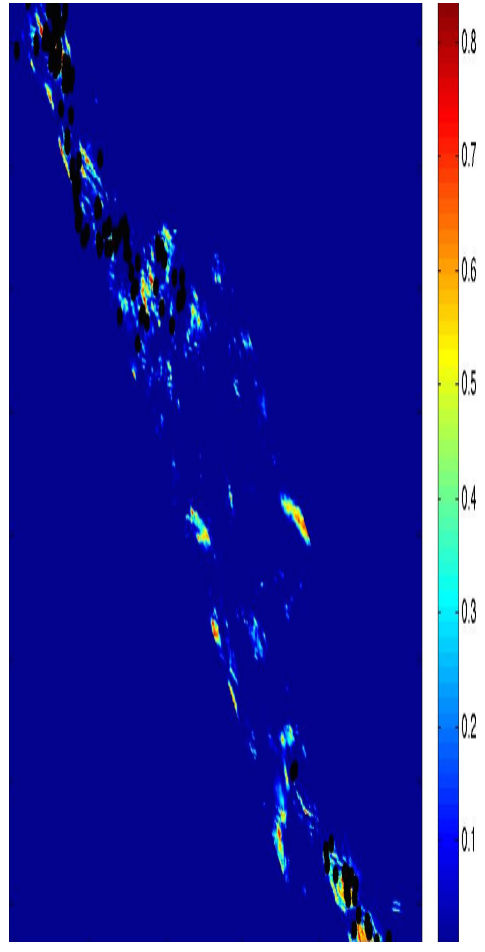












DNN - A quick overview

- Neural Networks with a particular (deep) architecture
- Exist for a long time (50's)
- For large amounts of data can perform very well
- Applications
  - Image classification
  - Face recognition
  - Segmentation
  - Driverless cars
  - ...

DNN - A quick overview

- Neural Networks with a particular (deep) architecture
- Exist for a long time (50's)
- For large amounts of data can perform very well
- Applications
  - Image classification
  - Face recognition
  - Segmentation
  - Driverless cars
  - ...
- A few recent quotes:
  - Apple Is Bringing the AI Revolution to Your iPhone, Wired, 2016
  - Why Deep Learning Is Suddenly Changing Your Life, Fortune, 2016
  - Google researchers are learning how machines

DNN - A quick overview

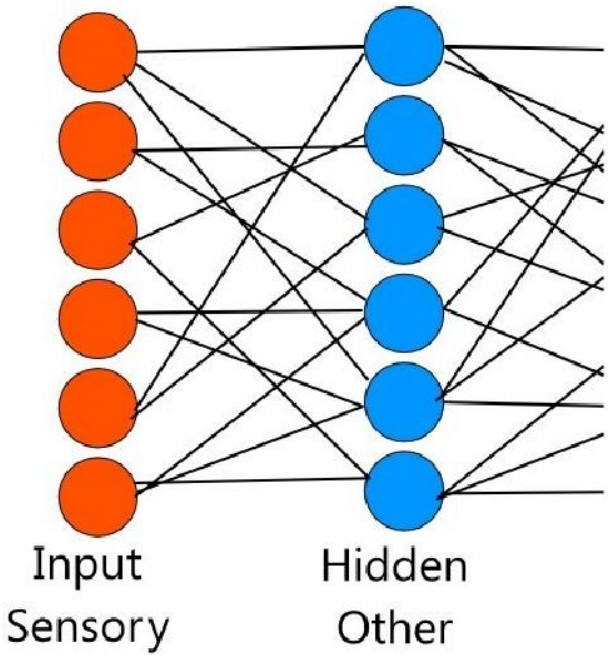
The Dark Secret at the Heart of AI - MIT Technology Review 2017

"No one really knows how the most advanced algorithms do what they do. That could be a problem.

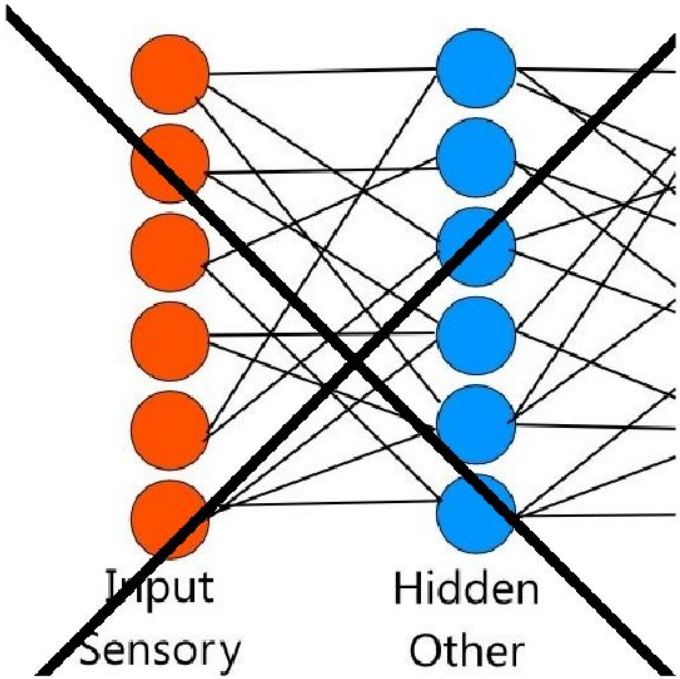
We can build these models, but we don't know how they work."

DNN - A quick overview

Data scientists point of view



DNN - A quick overview



Computational scientist point of view

$$\mathbf{Y}_{j+1} = \sigma(\mathbf{K}_j \mathbf{Y}_j + b_j)$$

DNN - ResNet

Residual networks - [He et-al], won prizes in 2015

$$\mathbf{Y}_{j+1} = \sigma(\mathbf{Y}_j + \mathbf{K}_j^{(2)}\sigma(\mathbf{K}_j^{(1)}\mathbf{Y}_j + b_j))$$

(small) modification

- Propagation: for $j = 1, \ldots, N$

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\mathbf{K}_j^{(2)}\sigma(\mathbf{K}_j^{(1)}\mathbf{Y}_j + b_j)$$

- Classification/Segmentation

$$\mathbf{C} \approx \mathbf{W}\,\mathbf{Y}_N$$

DNN - A quick overview

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\mathbf{K}_j^{(2)}\sigma(\mathbf{K}_j^{(1)}\mathbf{Y}_j + b_j) \qquad \mathbf{C} \approx \mathbf{W}\,\mathbf{Y}_N$$

- $\mathbf{Y}_1$ - data $\quad$ $\mathbf{C}$ - class
- $\mathbf{K}_j$ - weight matrix
- b_j - bias
- $\mathbf{W}$ - Classifier

Forward: Given an image $\mathbf{y}_1$ find its class $\mathbf{c}$

Inverse (training): Given data and classes $\{\mathbf{Y}_1, \mathbf{C}\}$ obtain $\mathbf{K}_j, b_j$ and $\mathbf{W}$ that approximately classify the given data

DNN - The optimization problem

Define a similarity measure $\mathcal{S}$ (e.g. cross entropy)

$$\min_{\mathbf{K}_j, b_j, \mathbf{W}} \quad \mathcal{S}(\mathbf{C}, \mathbf{W}\,\mathbf{Y}_N)$$

$$\text{s.t} \quad \mathbf{Y}_{j+1} = \mathbf{Y}_j + h\mathbf{K}_j^{(2)}\sigma(\mathbf{K}_j^{(1)}\mathbf{Y}_j + b_j)$$

Issues

- Instability and lack of generalization
- Many local solutions?
- Robustness

DNN as an ODE

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\mathbf{K}_j^{(2)}\sigma(\mathbf{K}_j^{(1)}\mathbf{Y}_j + b_j) \quad \leftrightarrow \quad \dot{\mathbf{Y}} = \mathbf{K}^{(2)}(t)\sigma(\mathbf{K}^{(1)}(t)\mathbf{Y} + b(t))$$

Path planning: Find a path that separate the different classes

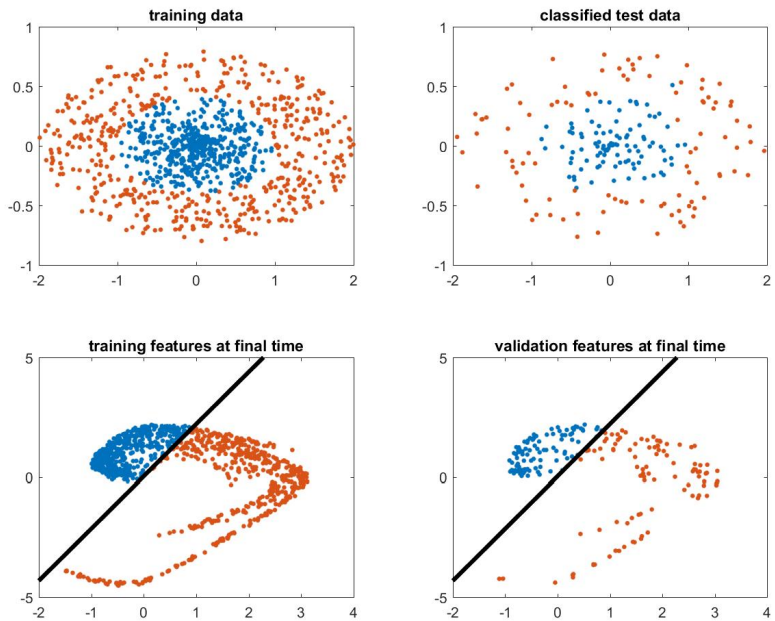
Dynamics movie

DNN as an ODE

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\mathbf{K}_j^{(2)}\sigma(\mathbf{K}_j^{(1)}\mathbf{Y}_j + b_j) \quad \leftrightarrow \quad \dot{\mathbf{Y}} = \mathbf{K}^{(2)}(t)\sigma(\mathbf{K}^{(1)}(t)\mathbf{Y} + b(t))$$

Path planning: Find a path that separate the different classes

Dynamics movie



DNN as an ODE

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\mathbf{K}_j^{(2)}\sigma(\mathbf{K}_j^{(1)}\mathbf{Y}_j + b_j) \quad \leftrightarrow \quad \dot{\mathbf{Y}} = \mathbf{K}^{(2)}(t)\sigma(\mathbf{K}^{(1)}(t)\mathbf{Y} + b(t))$$

Before we solve the learning problem we ask:
Is the forward (propagation) problem well posed?

- Is the ODE well posed?
- Is the discretized ODE well posed?

DNN as an ODE

Why is well-posedness important?

If we start from a perturb data we should classify in a similar way

stability movie

$$\dot{\mathbf{Y}} = \sigma(\mathbf{K}(t)\mathbf{Y} + b(t)) \quad \mathbf{Y}(t=0) = \mathbf{Y}_0 + \epsilon$$

Well posedness

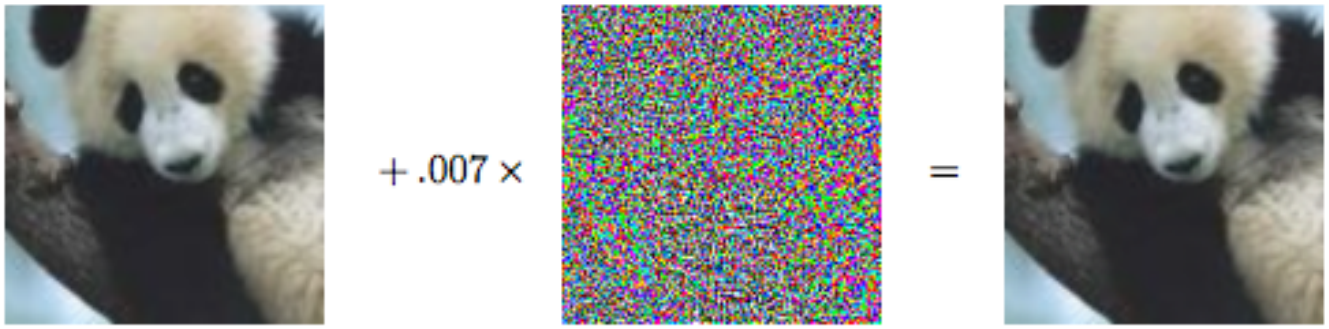
$$\|\mathbf{Y}(t=T, \mathbf{Y}_0) - \mathbf{Y}(t=T, \mathbf{Y}_0 + \epsilon)\|$$

is small

Stability and well-posedness

Symptoms of ill-posedness

- Exploding gradients
- Vanishing gradients
- Adversarial examples



Stability and well-posedness

Most DNN's are not well posed!

Stability and well-posedness

Most DNN's are not well posed!

How can we make them well-posed?

- Constrain the weight matrices
- Design new type of networks
- Make sure that the discrete dynamics is well posed

Stability and well-posedness

Constrain the weight matrices

(A very old) Theorem
The ODE $\dot{\mathbf{y}} = f(\mathbf{y})$ is well posed if

- The real part of the eigenvalues of the Jacobian of f are not positive.
- The Jacobain is changing slowly in time.
- Can be better characterized by the **kinematic eigenvalues** of the Jacobain (Ascher [84])

In the DNN case

$$\mathbf{J}(\mathbf{Y}) = \mathbf{K}^{(2)}(t)\mathrm{diag}(\sigma')\mathbf{K}^{(1)}(t)$$

Stability and well-posedness

Choosing $\mathbf{K}^{(2)} = \mathbf{I}$ we have that:
If $Re(\mathrm{eig}(\mathbf{K})) \leq 0$ the (continuous) DNN is well-posed

Stability and well-posedness

Choosing $\mathbf{K}^{(2)} = \mathbf{I}$ we have that:
If $Re(\mathrm{eig}(\mathbf{K})) \leq 0$ the (continuous) DNN is well-posed

Constrain the weight matrices

Unconditionally stable network

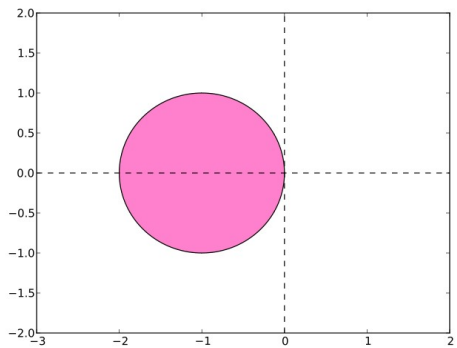
$$\dot{\mathbf{Y}} = \sigma((\mathbf{A}(t) - \mathbf{A}(t)^\top)\mathbf{Y} + b(t))$$

$$\begin{pmatrix} \dot{\mathbf{Y}} \\ \mathbf{Z} \end{pmatrix} = \sigma\left(\begin{pmatrix} 0 & \mathbf{A}(t)^\top \\ -\mathbf{A}(t) & 0 \end{pmatrix}\begin{pmatrix} \mathbf{Y} \\ \mathbf{Z} \end{pmatrix} + b(t)\right)$$

Stability and well-posedness

Discrete stability

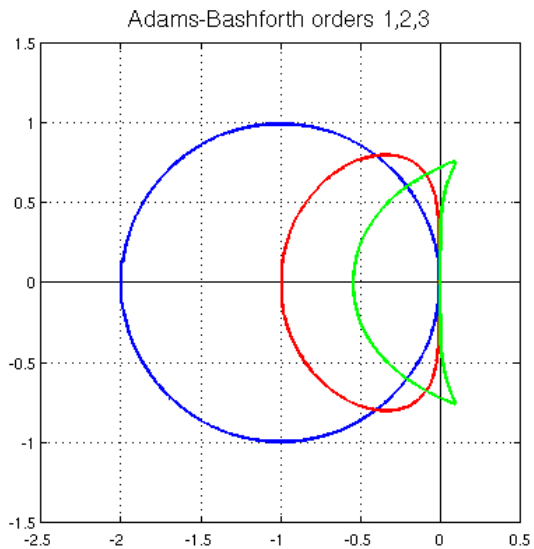
The classical resNet is **unstable** for purely imaginary eigenvalues



Stability and well-posedness

Discrete stability - Adam Bashford 3 network

$$\begin{aligned}\mathbf{Y}_{n+1} &= \mathbf{Y}_n + \frac{h}{12}\left(23\sigma(\mathbf{K}_n\mathbf{Y}_n + b_n) - \right.\\ &\quad 16\sigma(\mathbf{K}_{n-1}\mathbf{Y}_{n-1} + b_{n-1}) + \\ &\quad \left. 5\sigma(\mathbf{K}_{n-2}\mathbf{Y}_{n-2} + b_{n-2})\right)\end{aligned}$$



Stability and well-posedness

Discrete stability - Hamiltonian inspired networks
Verlet Network

$$\begin{aligned} \mathbf{Y}_{n+1} &= \mathbf{Y}_n + h\sigma(\mathbf{A}_n\mathbf{Z}_{n+\frac{1}{2}} + b_n) \\ \mathbf{Z}_{n+\frac{3}{2}} &= \mathbf{Z}_{n+\frac{1}{2}} - h\sigma(\mathbf{A}_n^\top\mathbf{Y}_{n+1} + b_n) \end{aligned}$$

Symplectic integration

Explicit methods - need to make sure h is small enough

More networks

Zoo of ODE solvers $\leftrightarrow$ network architecture

RK methods, multistep methods, implicit? ...

Pick your favorite ODE solver ...

Discretize $\Longleftrightarrow$ Optimize

We have an optimization problem

$$\min_{\theta} \; f(\theta; \mathbf{y}) \qquad \text{s.t } \dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, \theta)$$

Two options

- Discretize the ode first and solve a discrete optimization problem. (DO)
- Write the conditions for optimization in infinite space and then optimize. (OD)

Discretize $\Longleftrightarrow$ Optimize

Example:

$$\min_{\theta} \frac{1}{2}(y(1) - c)^2 \qquad \text{s.t } \dot{y} = -y \quad y_0 = \theta$$

The Lagrangian

$$\mathcal{L} = \frac{1}{2}(y(1) - c)^2 + \int_0^1 u(t)(\dot{y} + y)dt + u_0(y_0 - \theta)$$

Discretize $\Longleftrightarrow$ Optimize

Conditions for a min

$$\dot{y} + y = 0 \qquad y_0 - \theta = 0$$
$$\dot{u} - u = 0 \qquad u_1 = y(1) - c$$

The equations are **continuous** adjoints of each other

But

If we discretize using forward Euler with nonuniform steps then they are not **discrete adjoints** of each other

The discrete gradient may not be a descent direction

Discretize $\Longleftrightarrow$ Optimize

If we discretize first then optimize then these problems do not arise.

Need to make sure the discretization is somewhat faithful to the ODE

Optimization

For the computation of gradient use the adjoint method

Adjoint = Back Propagation (Without all the hype)

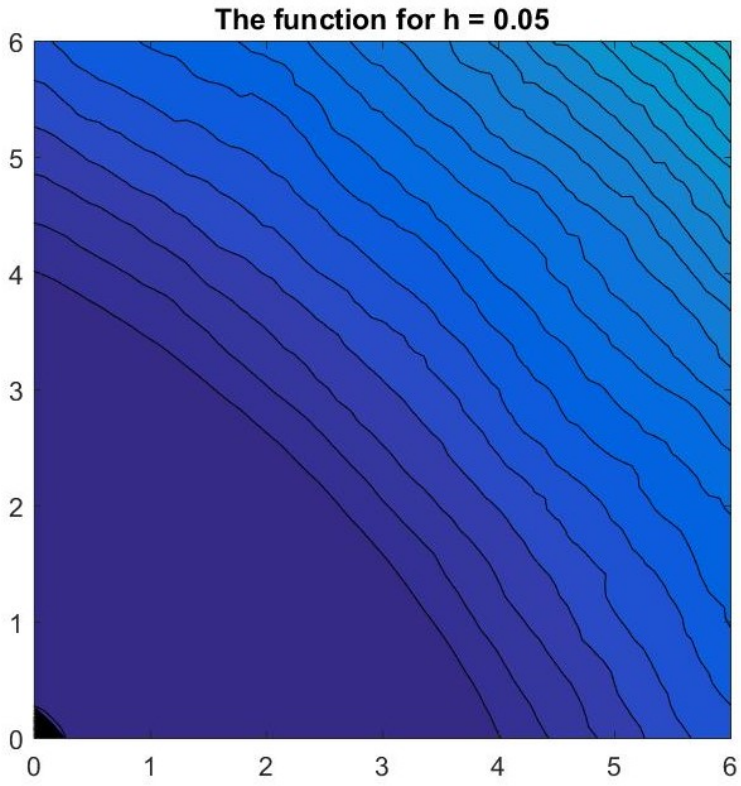
Optimization engine, Stochastic Gradient Descent (SGD)

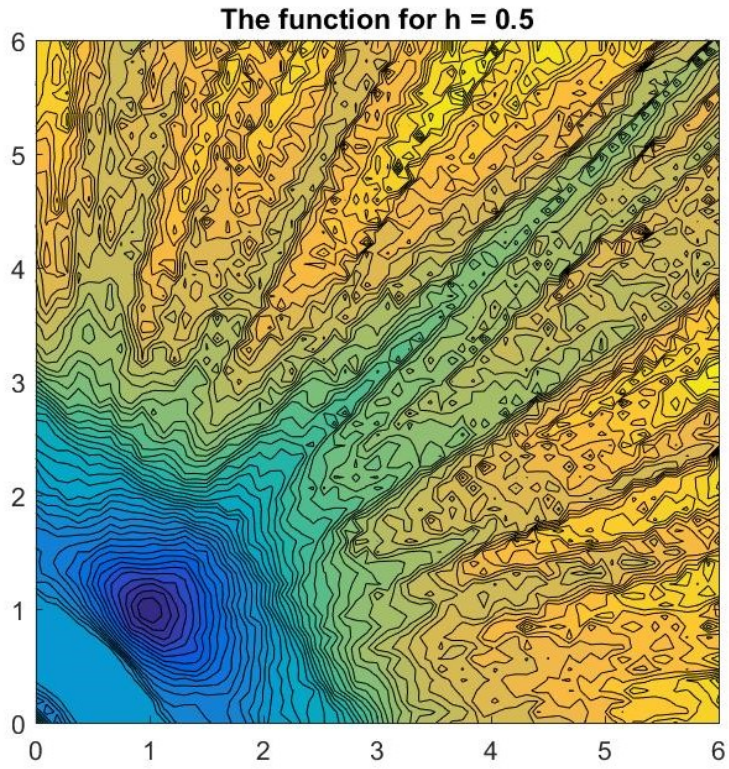
- SGD vs Stochastic Avererage Approximation
- Inexact Gauss-Newton
- No need to compute Hessians only mat-vecs
- No problems of exploding/vanishing gradients

Why should you care about stability

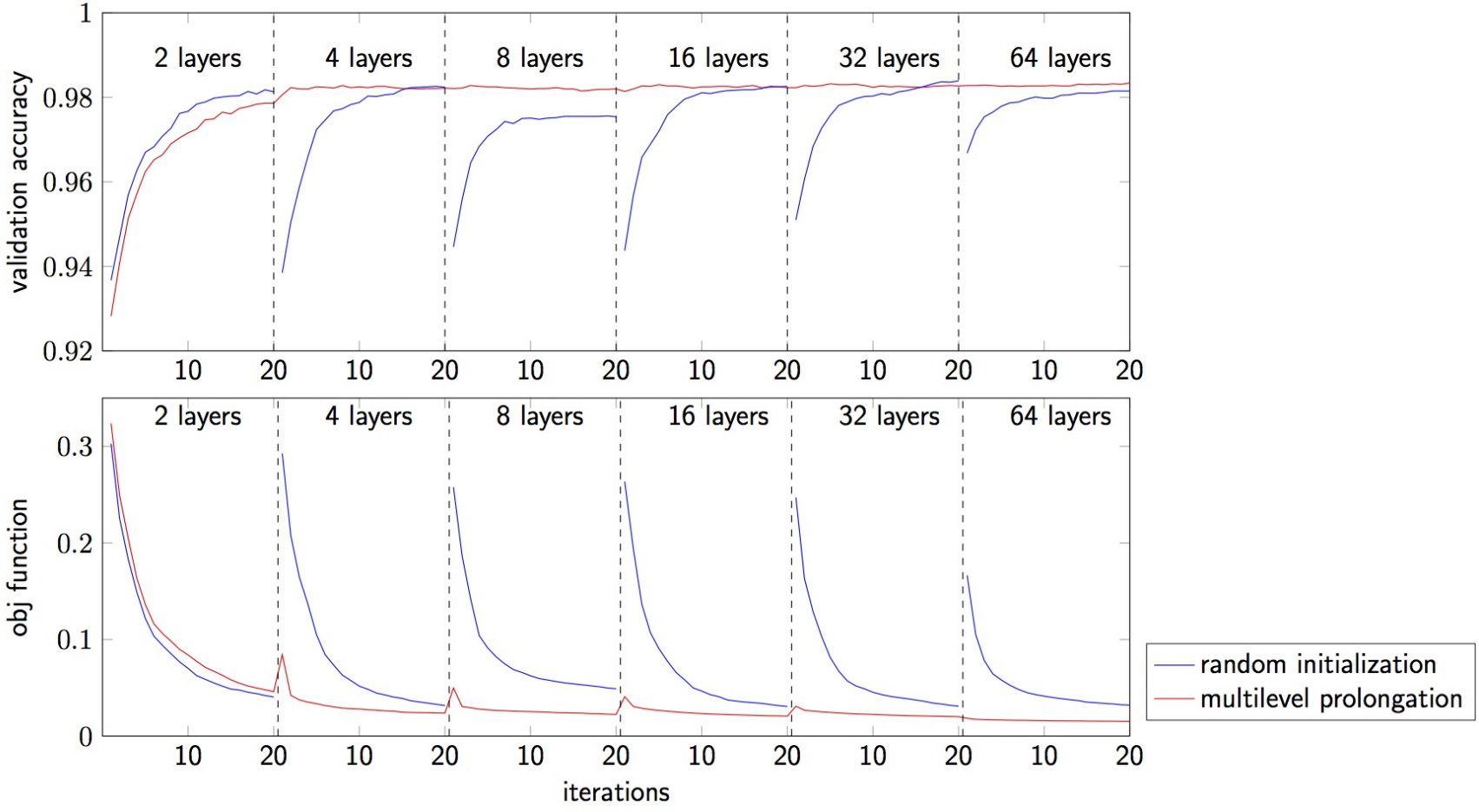
$$\min \frac{1}{2}\|\mathbf{Y}_N(\theta_1.\theta_2) - \mathbf{C}\|^2 \qquad \mathbf{Y}_{j+1} = \mathbf{Y}_j - h\mathbf{K}^\top \sigma\left(\mathbf{K}\mathbf{Y}_j\right)$$

$$\mathbf{K} = \begin{pmatrix} \theta_1 + \theta_2 & -\theta_1 & -\theta_2 \\ -\theta_2 & \theta_1 + \theta_2 & -\theta_1 \\ -\theta_1 & -\theta_2 & \theta_1 + \theta_2 \end{pmatrix}$$





Application to training - layer continuation



Time continuation

Convolution Neural Networks - A quick overview

- For large dimension data point not practical to have dense $\mathbf{K}$
- If $\mathbf{Y}$ is an image use sparse $\mathbf{K}$ - convolution

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\mathbf{K}_j^{(2)}\sigma(\mathbf{K}_j^{(1)}\mathbf{Y}_j + b_j)$$

- $\mathbf{Y}_1$ - data $\quad$ $\mathbf{c}$ - class
- $\mathbf{K}_j$ - convolution kernel
- b_j - bias

Convolution and PDE's

1D convolution

$$\mathbf{Ky} = [\mathbf{K}_1, \mathbf{K}_2, \mathbf{K}_3] * [\mathbf{y}_1, \ldots, \mathbf{y}_n]$$

Change the basis

$$\mathbf{Ky} = \mathbf{s}_1 * [0, 1, 0] + \frac{\mathbf{s}_2}{2h} * [-1, 0, 1] + \frac{\mathbf{s}_3}{h^2}[1, -2, 1]$$

$\mathbf{y}$ - a discretization (grid function) of $y(x)$
At the limit $h \to 0$

$$\mathbf{Ky} \approx \mathbf{s}_1 y + \mathbf{s}_2 \frac{dy}{dx} + \mathbf{s}_3 \frac{d^2 y}{dx^2}$$

Convolution and PDE's

The convolution model

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\mathbf{K}_j^{(2)}\sigma(\mathbf{K}_j^{(1)}\mathbf{Y}_j + b_j)$$

Choose

$$\mathbf{K}_j^{(2)} = -(\mathbf{K}_j^{(1)})^\top$$

obtain a discretization of the **parabolic** PDE

$$\mathbf{Y}_t = -\mathbf{K}(t)^\top\sigma(\mathbf{K}(t)\mathbf{Y} + b(t))$$

Parabolic Networks

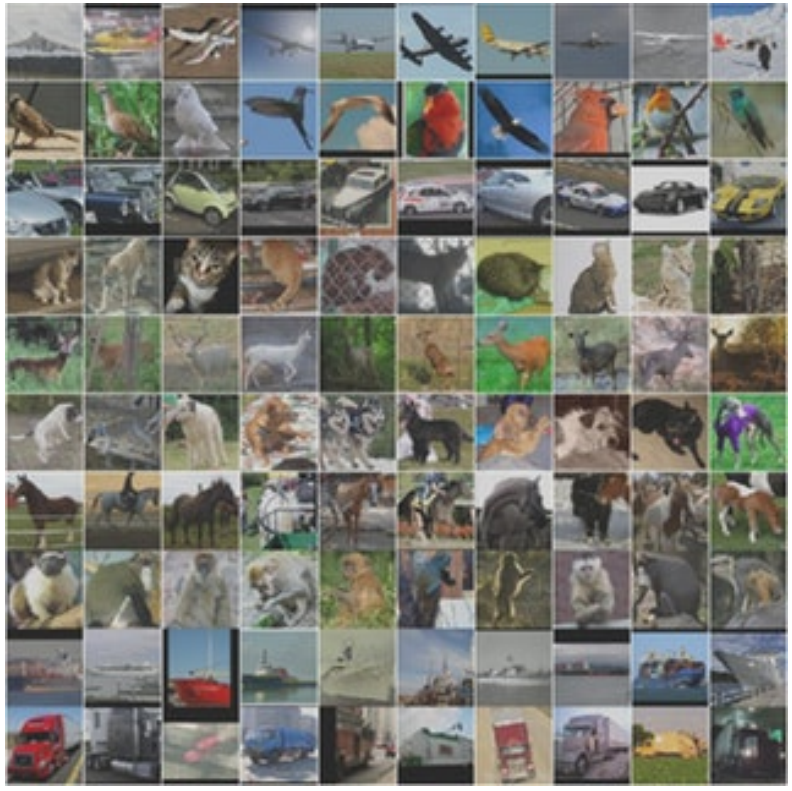
The convolution model

$$\mathbf{Y}_t = -\mathbf{K}(t)^\top \sigma(\mathbf{K}(t)\mathbf{Y} + b(t))$$

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j - h\mathbf{K}_j^\top \sigma(\mathbf{K}_j\mathbf{Y}_j + b_j)$$

- CNN can be interpreted as anisotropic diffusion that is very common in image processing, filtering, segmentation, etc.
- Enable learning the best operator based on the examples

Experiment - STL-10



Google	Intel	Chinese UHK	Nvidia	Facebook	Xtract
2013	2014	2015	2015	2016	2017
70.2%	72.8%	73.15%	74.10%	74.33%	83.20%

Hyperbolic Networks

The convolution model

$$\mathbf{Y}_{tt} = -\mathbf{K}(t)^\top \sigma(\mathbf{K}(t)\mathbf{Y} + b(t))$$

$$\mathbf{Y}_{j+1} = 2\mathbf{Y}_j - \mathbf{Y}_{j-1} - h^2 \mathbf{K}_j^\top \sigma(\mathbf{K}_j \mathbf{Y} + b_j)$$

- Reversible architecture
- Requires much less memory

Implicit method

- Keeping stability can be difficult
- Field of view - explicit methods have a limited view

Implicit (but not practical)

$$\mathbf{Y}_{j+1} - \mathbf{Y}_j = h\sigma(\mathbf{K}_{j+1}\mathbf{Y}_{j+1})$$

Making it practical, semi-implicit and diffusion reaction methods

Diffusion Reaction Networks

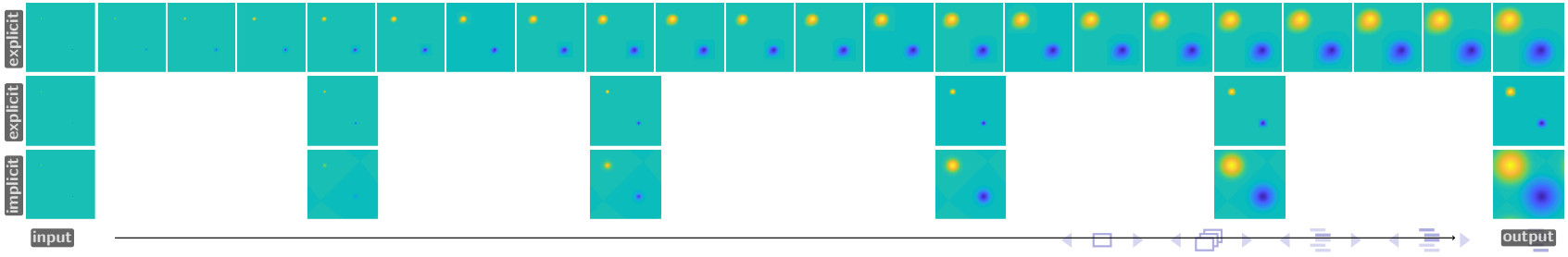
The convolution model

$$\mathbf{Y}_t = -\mathbf{K}(t)^\top \mathbf{K}(t)\mathbf{Y} + \sigma(\mathbf{S}(t)\mathbf{Y} + b(t))$$

Use IMEX discretization

$$\begin{aligned} \mathbf{Y}_{j+\frac{1}{2}} &= \mathbf{Y}_j + h\sigma(\mathbf{S}_j\mathbf{Y}_j + b_j) \\ \mathbf{Y}_{j+1} &= (\mathbf{I} + h\mathbf{K}_j^\top \mathbf{K}_j)^{-1}\mathbf{Y}_{j+\frac{1}{2}} \end{aligned}$$

Steps have non-local effects

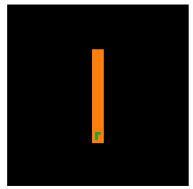
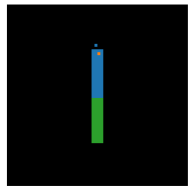
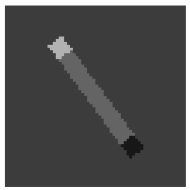
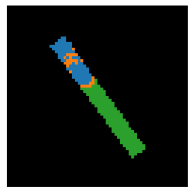
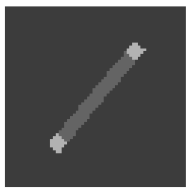
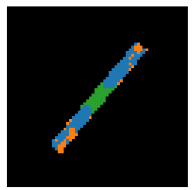
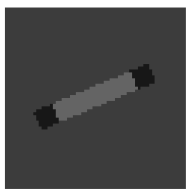
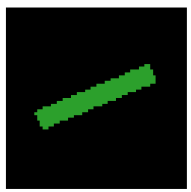
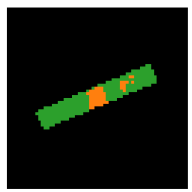


Sparse vs Dense tasks

- In sparse task we go from the data to a lower dimensional manifold (e.g. classification)
- On dense tasks we stay within the same/increase dimension (e.g. segmentation)

Memory requirements makes standard algorithms fail for most large-scale 3D data sets.

Segmentation - the Q-tips data set

Image	Segmentation	IMEX Pred	ResNet Pred
			
			
			
			

Segmentation - the Q-tips data set

Network	Parameters	IOU	Loss	Accuracy
IMEXnet	2701440	0.926	0.0982	99.56
ResNet	2691648	0.741	0.3332	98.18

- IMEX does better than ResNet
- Insignificant addition of parameters and computation
- Gain stability and field of view

Segmentation - CAMVID

	Accuracy	IoU	MeanBFScore
Sky	0.95	0.91	0.92
Building	0.81	0.77	0.64
Pole	0.70	0.10	0.54
Road	0.95	0.91	0.78
Pavement	0.91	0.74	0.74
Tree	0.89	0.78	0.69
SignSymbol	0.79	0.38	0.49
Fence	0.81	0.53	0.52
Car	0.93	0.79	0.74
Pedestrian	0.87	0.34	0.41
Bicyclist	0.82	0.51	0.42

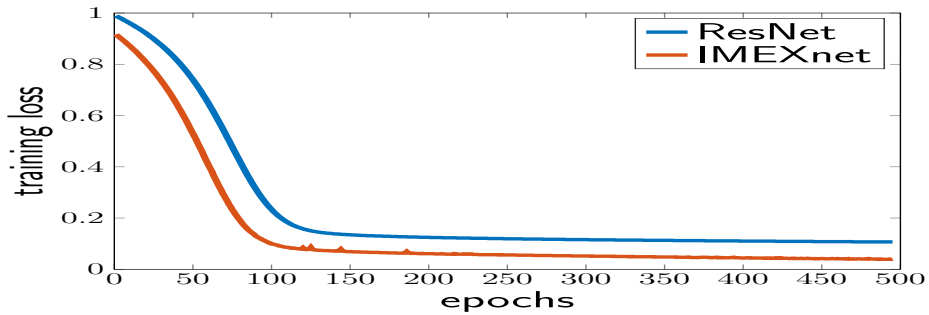
Overall within 0.5% of latest published results (Jan 2018) with 50% of parameters.

Depth prediction using a single image

Goal - predict the depth of the kitchen

Depth prediction using a single image

Convergence of training



Kitchen scene

Depth map

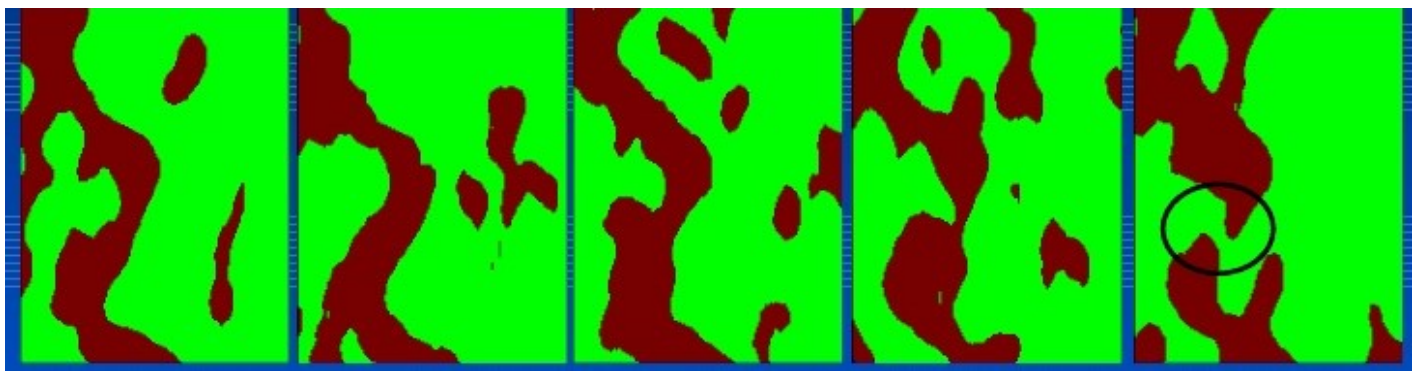
AI in Geoscience

- Data size
- Data accuracy and noise
- Sampling issues
- Data understanding

Data size issues

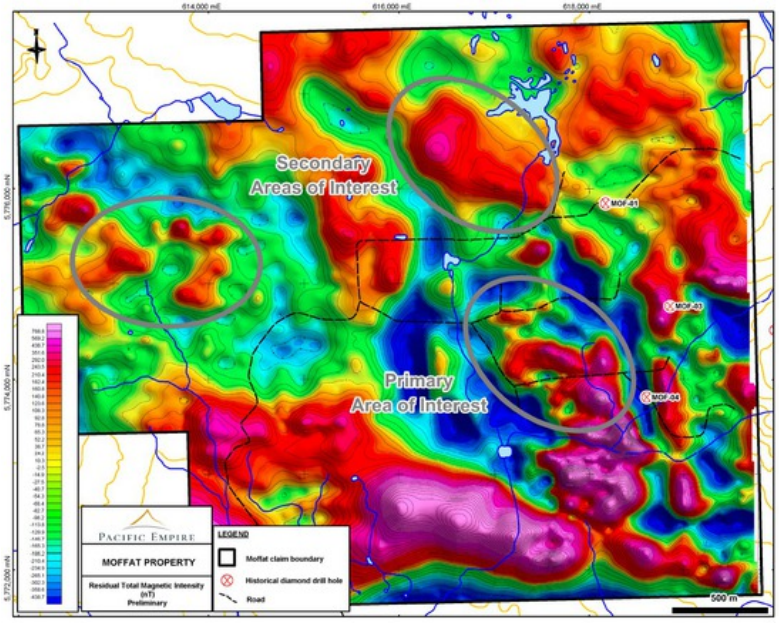
Most geophysical data includes only a few images.
Need sophisticated data augmentation methods

- Multi Point Statistics (MPS)
- Simulated data
- GANS



Data accuracy

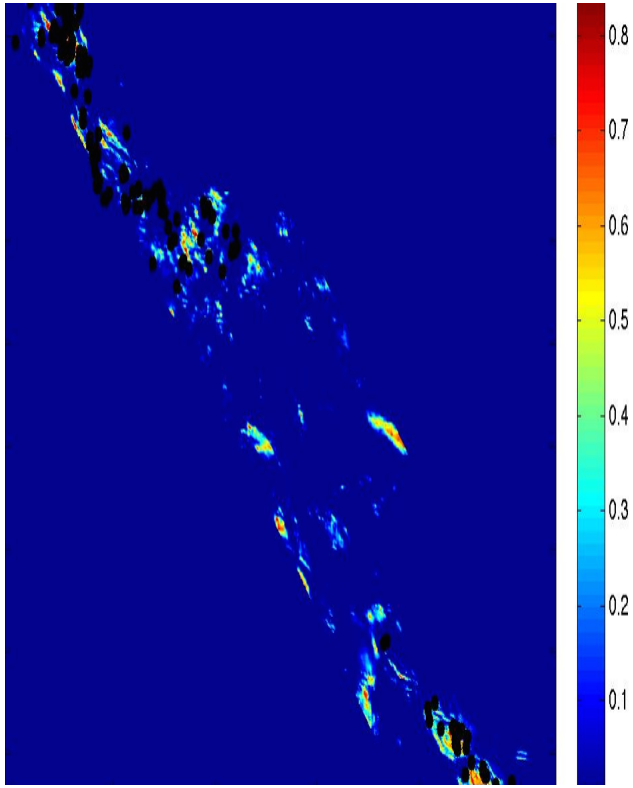
Cat



Porphyry???

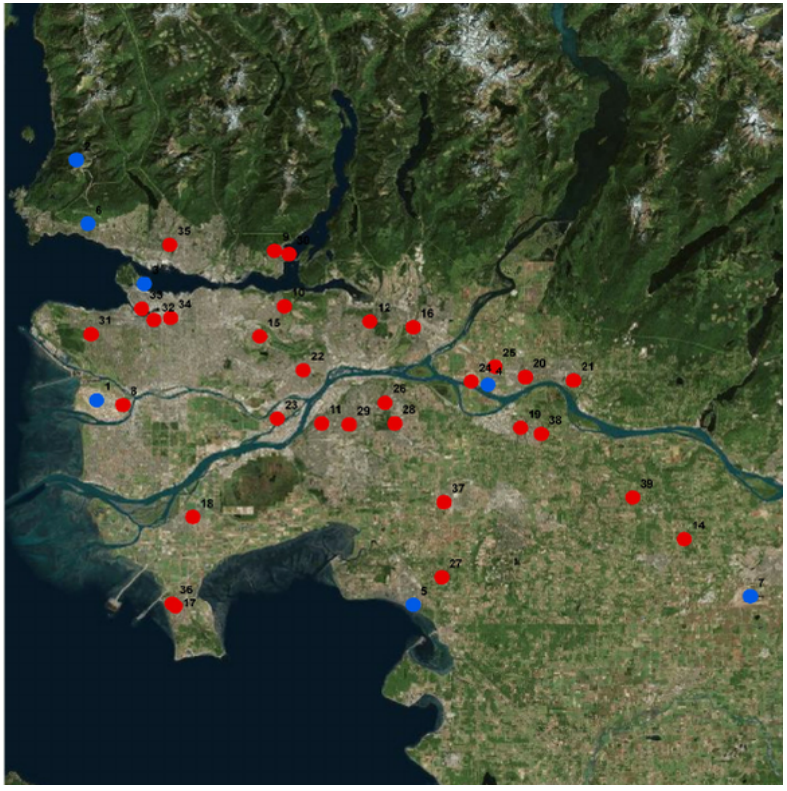
Need to add standard deviations to labels

Data sparsity and imbalance

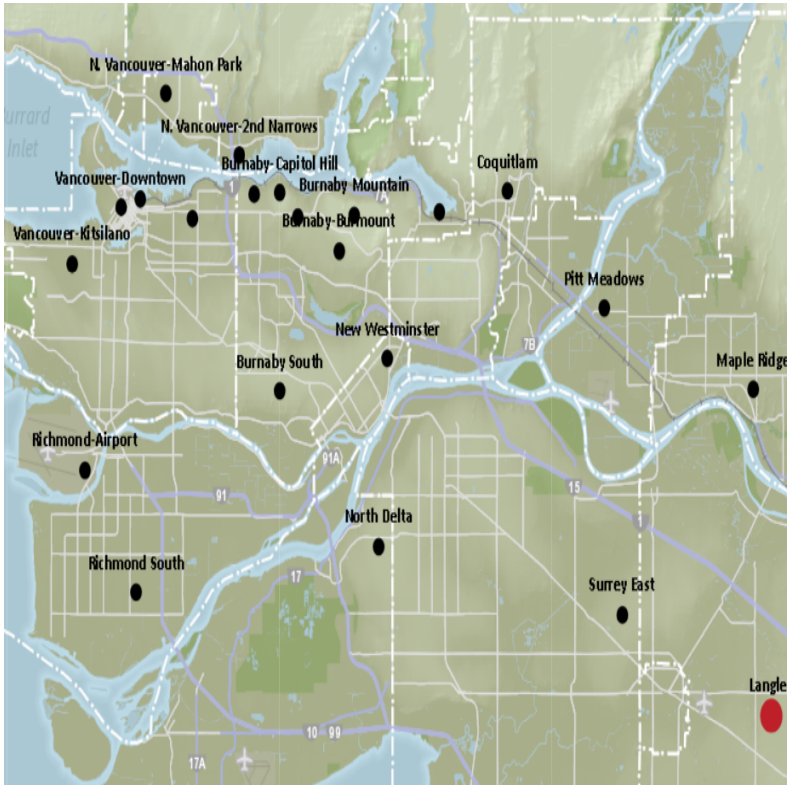


We know what is positive label but do not know how to label most of the image

Sampling issues



Weather stations



Air quality stations

Data is sampled on a non-uniform grid - CNN works on a uniform grid

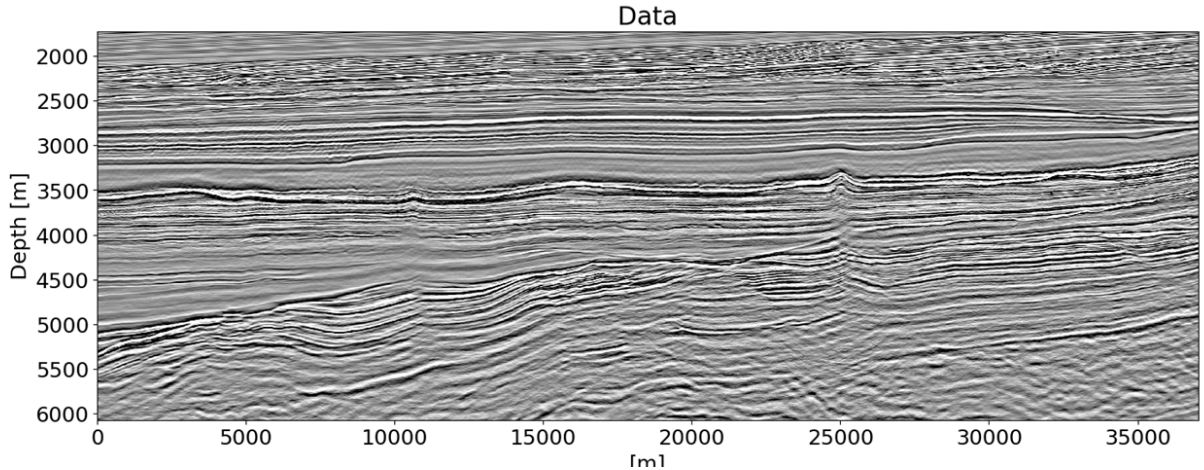
AI for geoscience

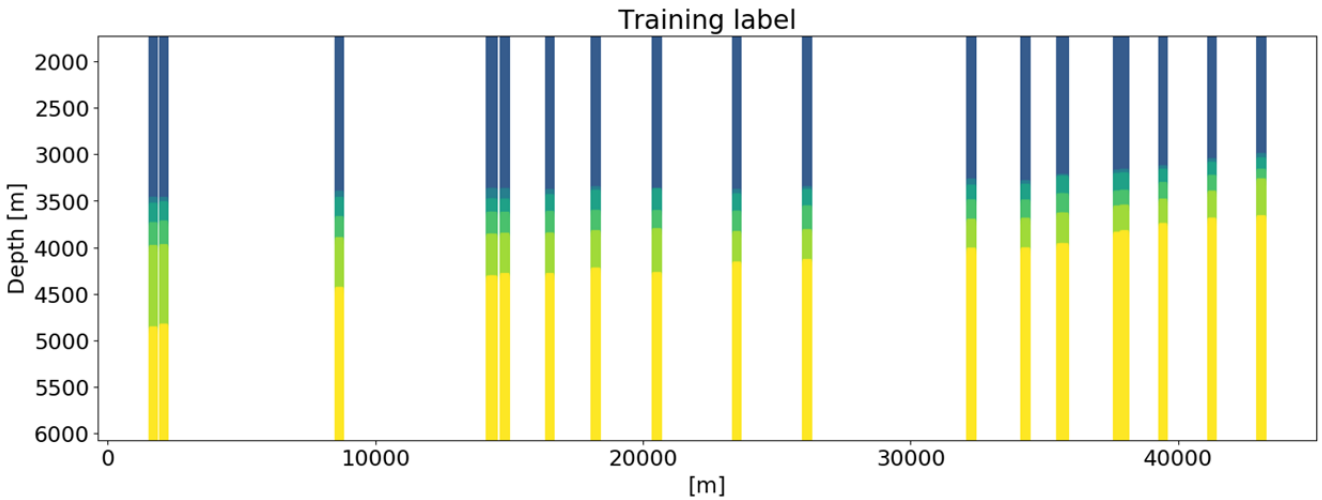
Need to rethink many of the ML algorithms

Adaptation and new ideas to deal with these difficulties

Need to have community data sets and test problems

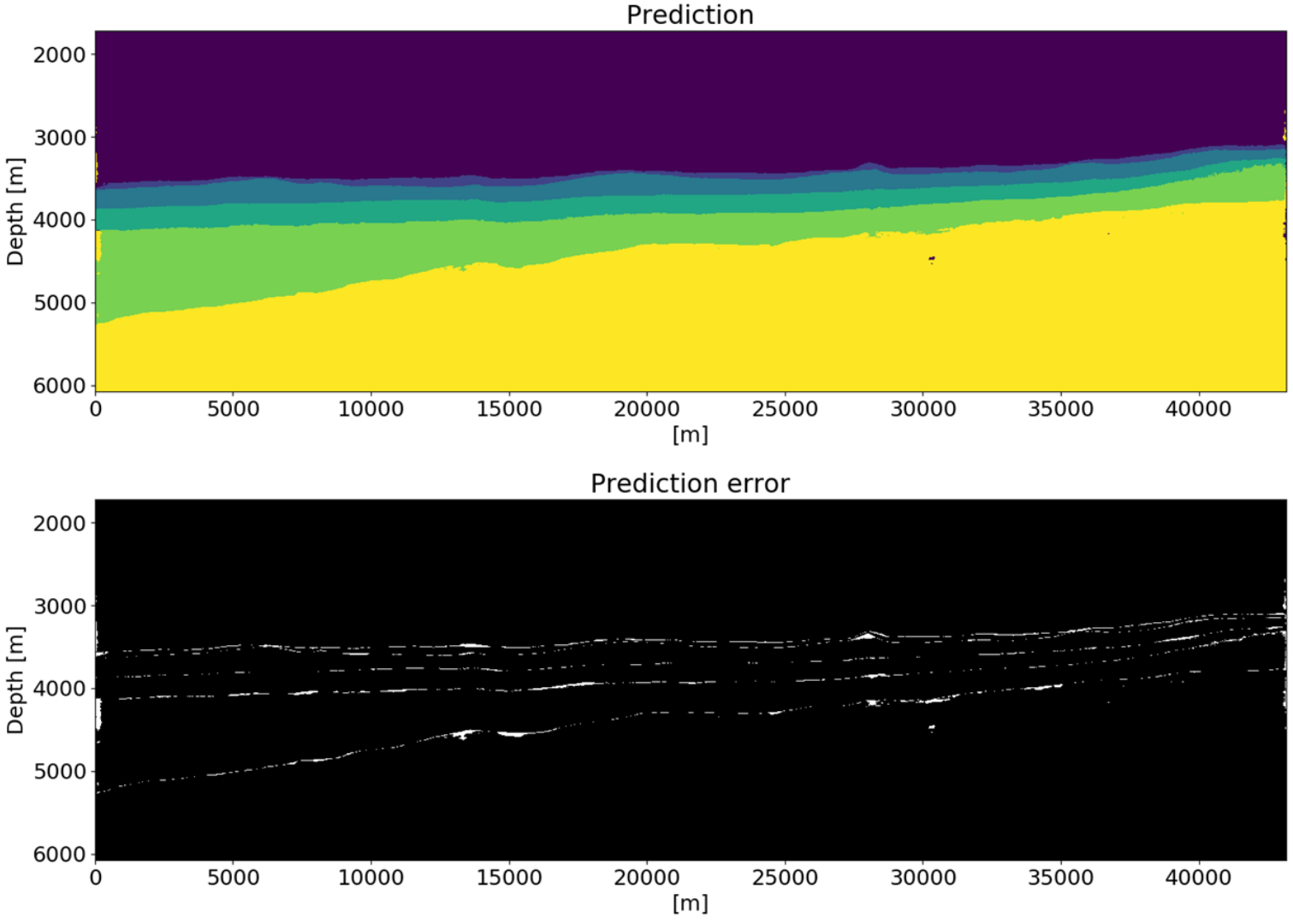
Lithology interpolation



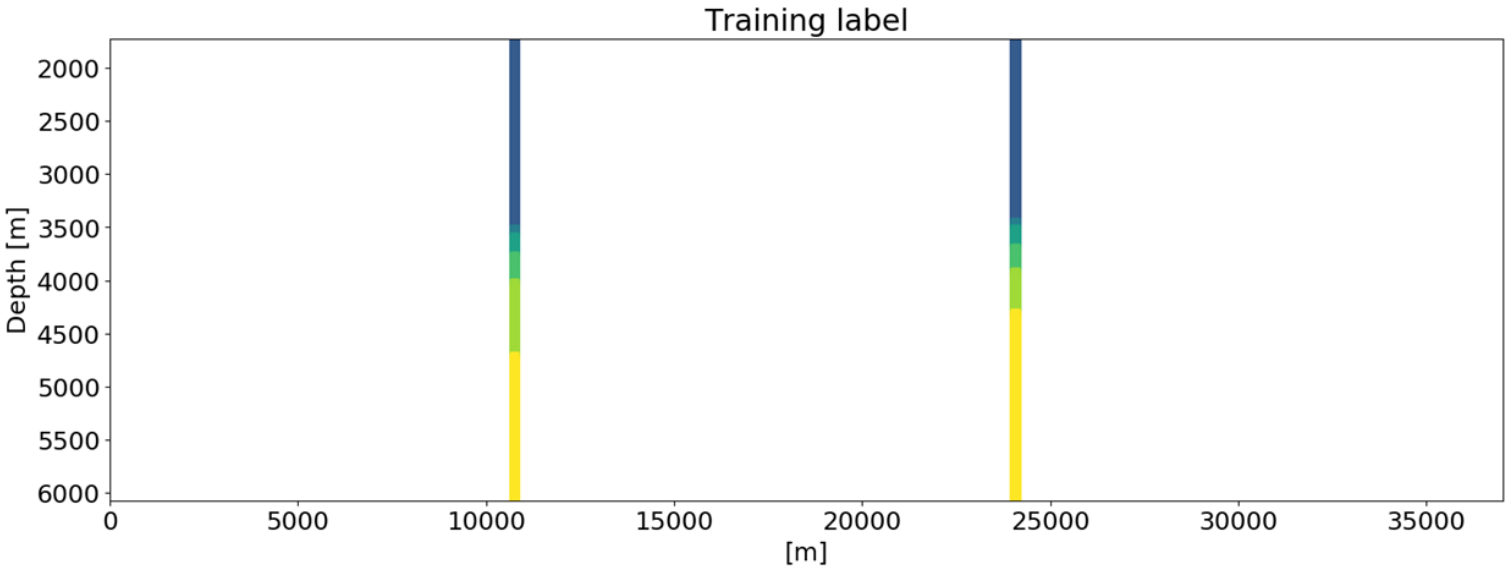


- Suppose we have the lithology information from many wells or manual interpretation

Lithology interpolation

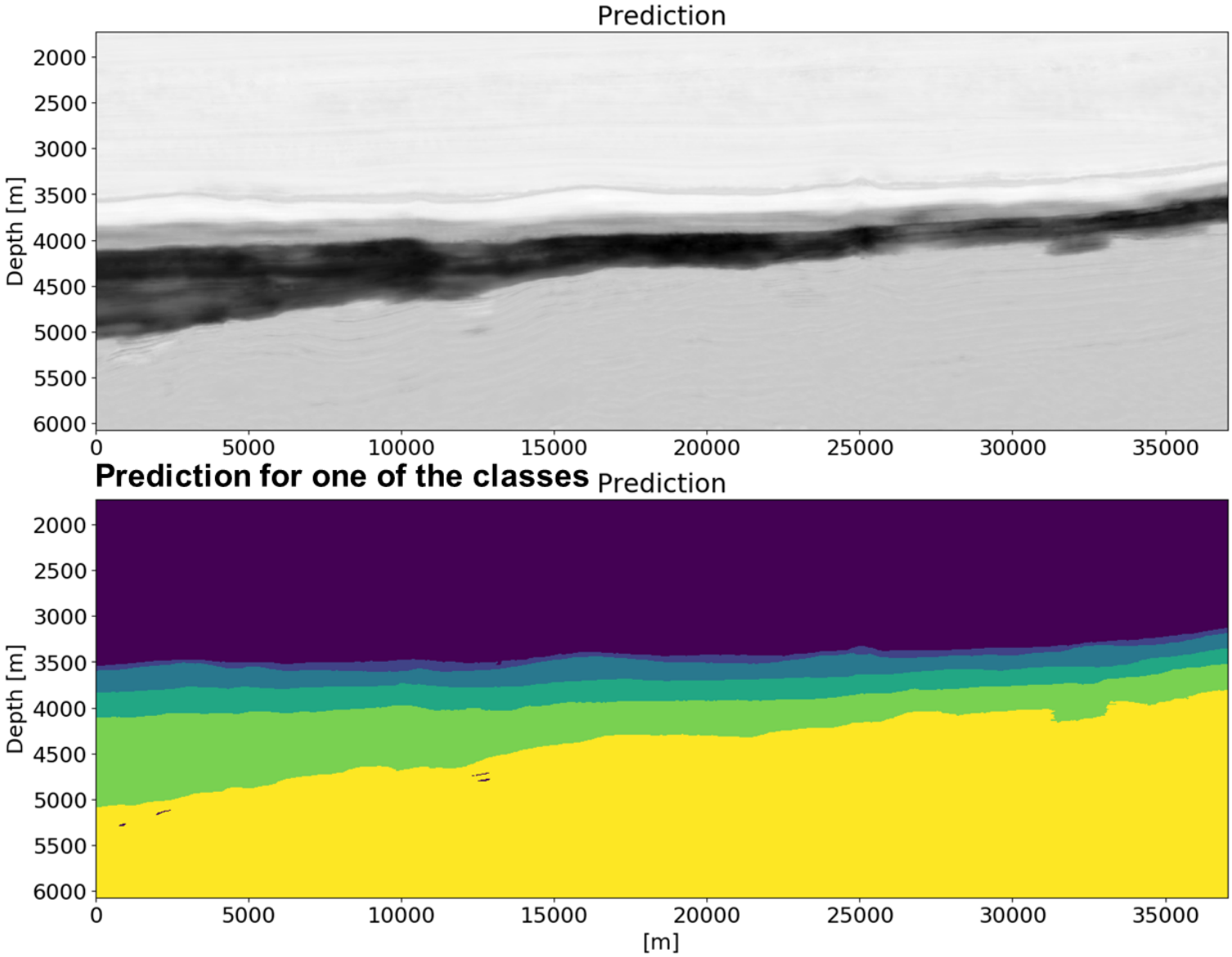


Lithology interpolation

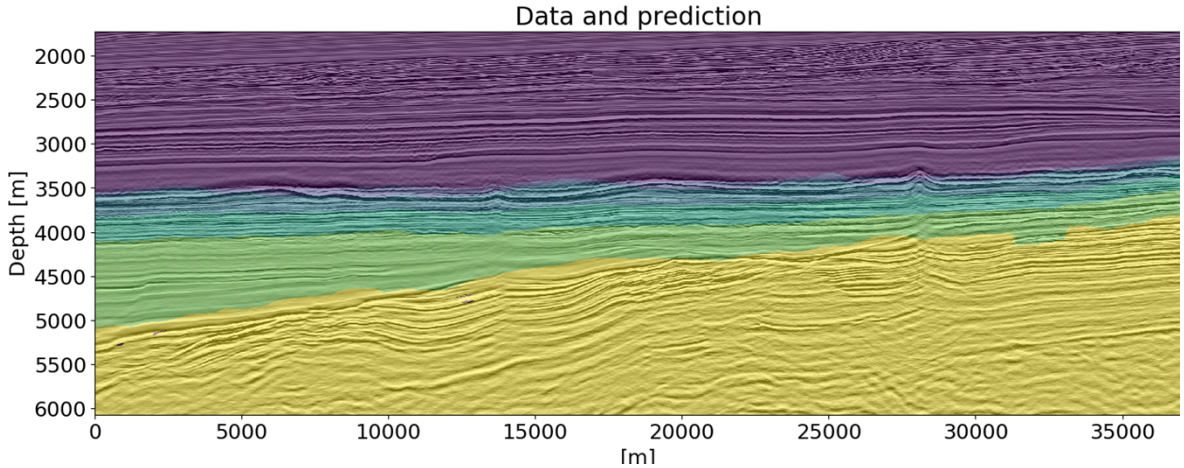


- What if we have only information from a few wells?
- 24 slices, 2 labeled wells each

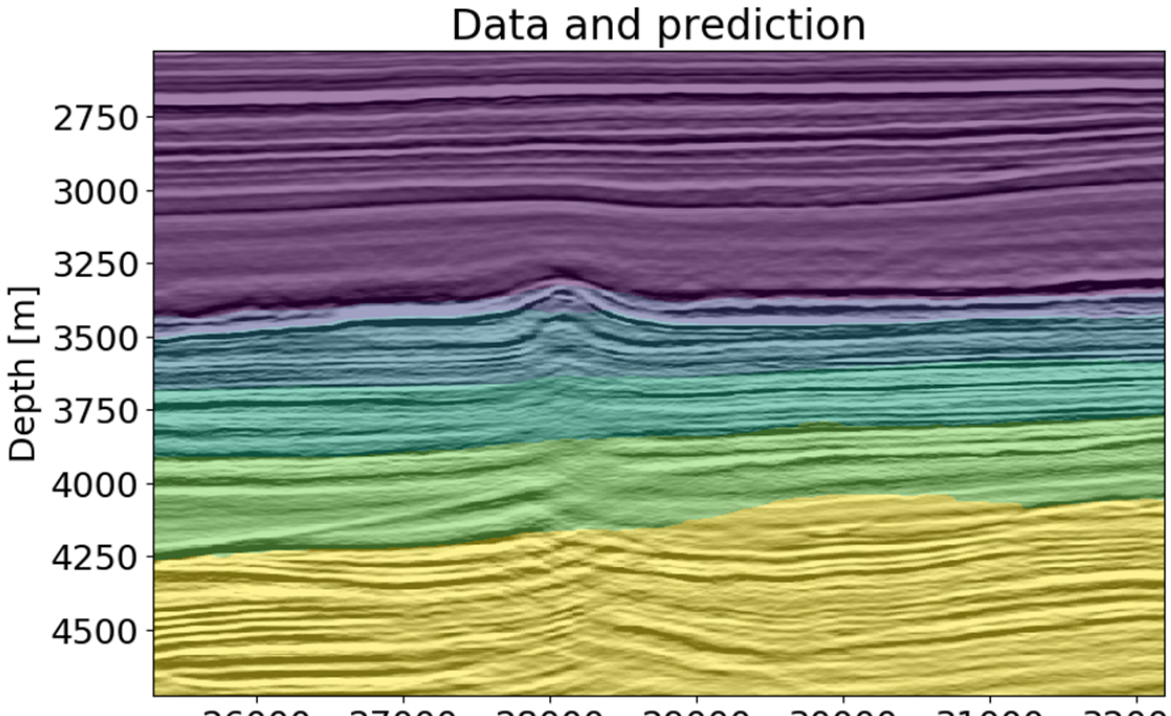
Lithology interpolation



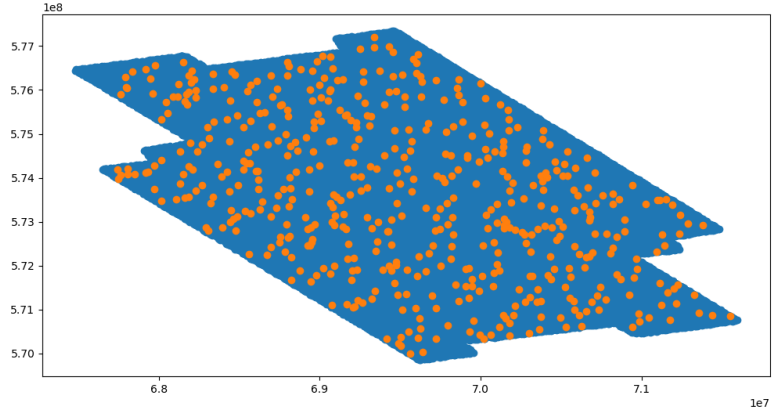
Lithology interpolation

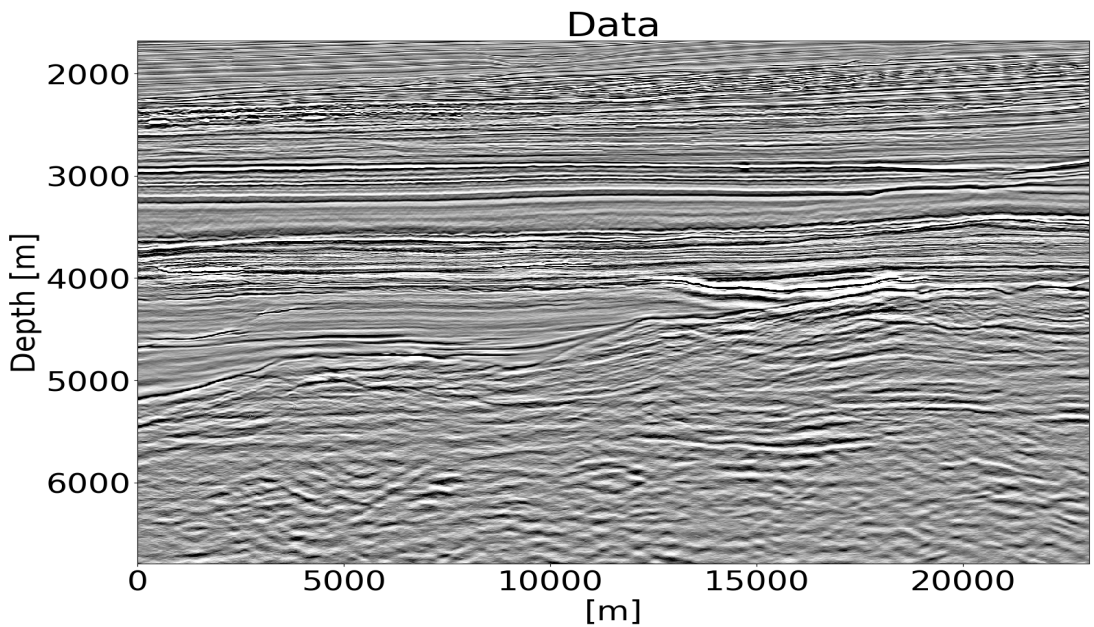


Zoomed

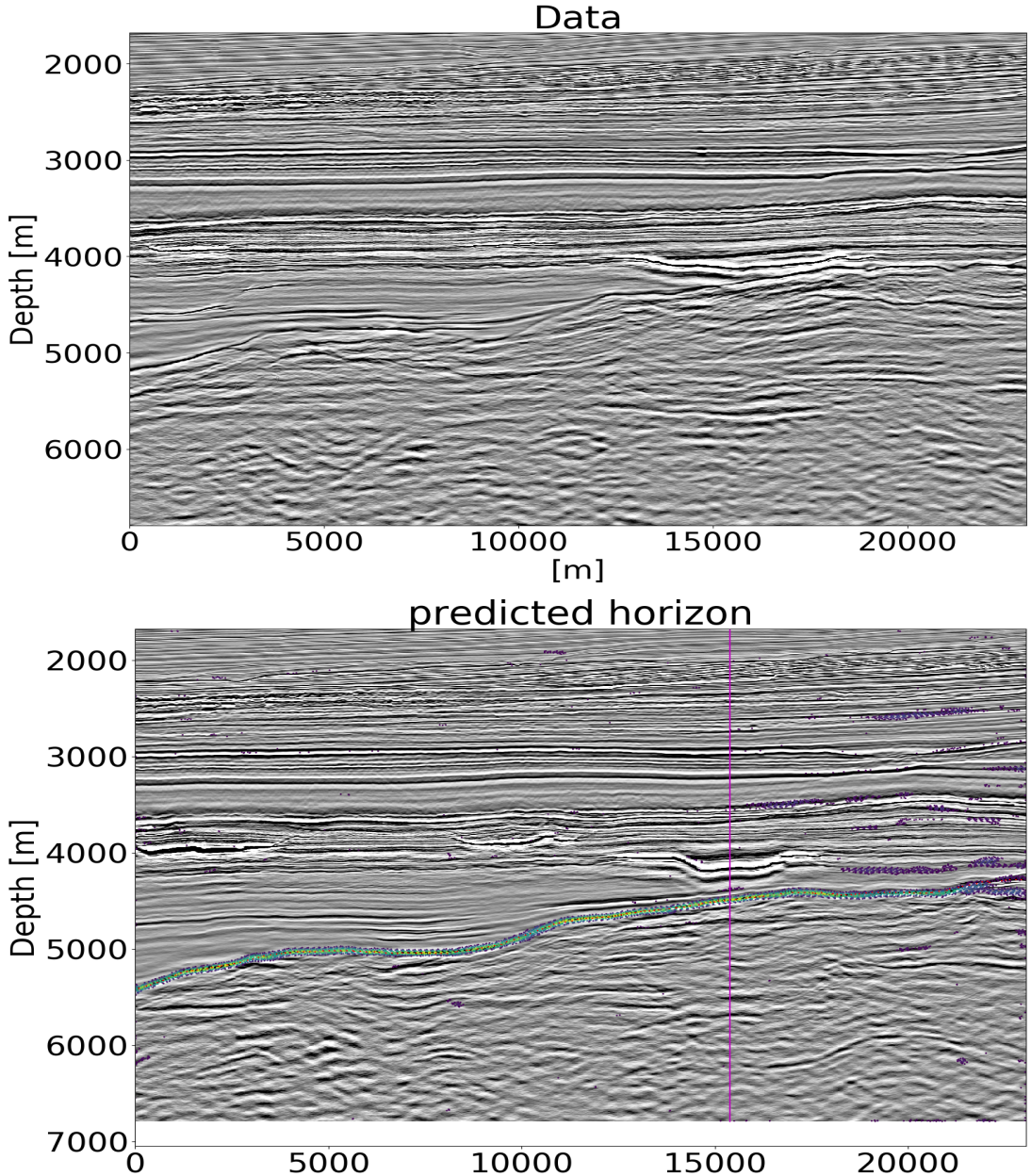


Picking horizons in 3D

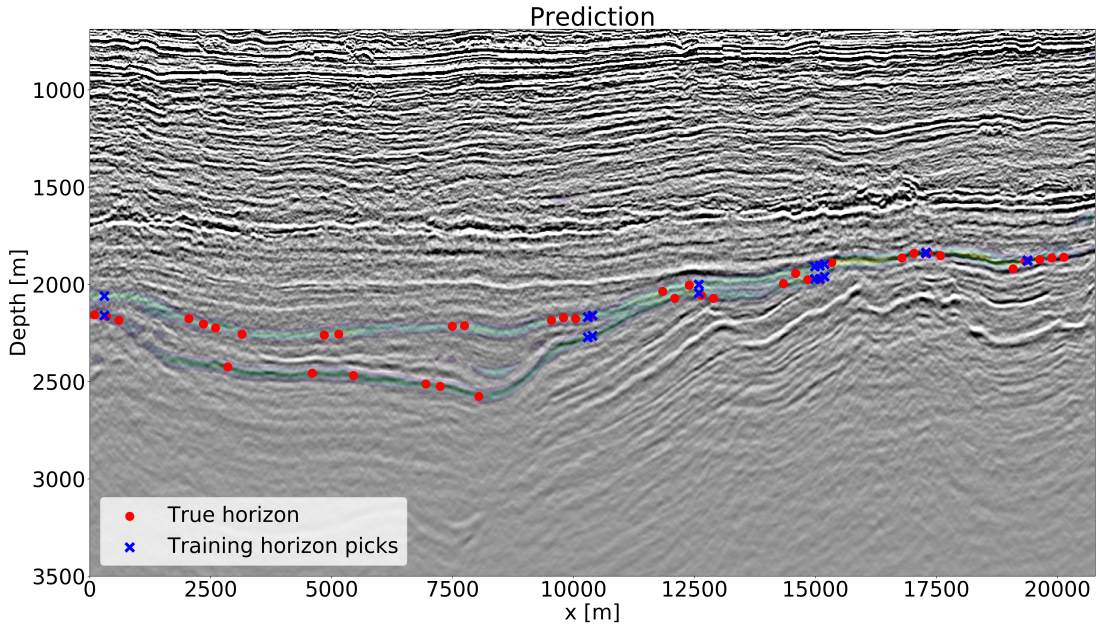




Picking horizons in 3D



Picking horizons in 3D



Can deal with forking

Conclusions

- The learning problem can be cast as a PDE constrained optimization problem
- Choose the PDE based on the task
- Extension of known techniques such as anisotropic filtering
- Need special care when applying to geoscience
- Success on Seismic Semantic Segmentation with limited data
- Better than state of the art for horizon picking