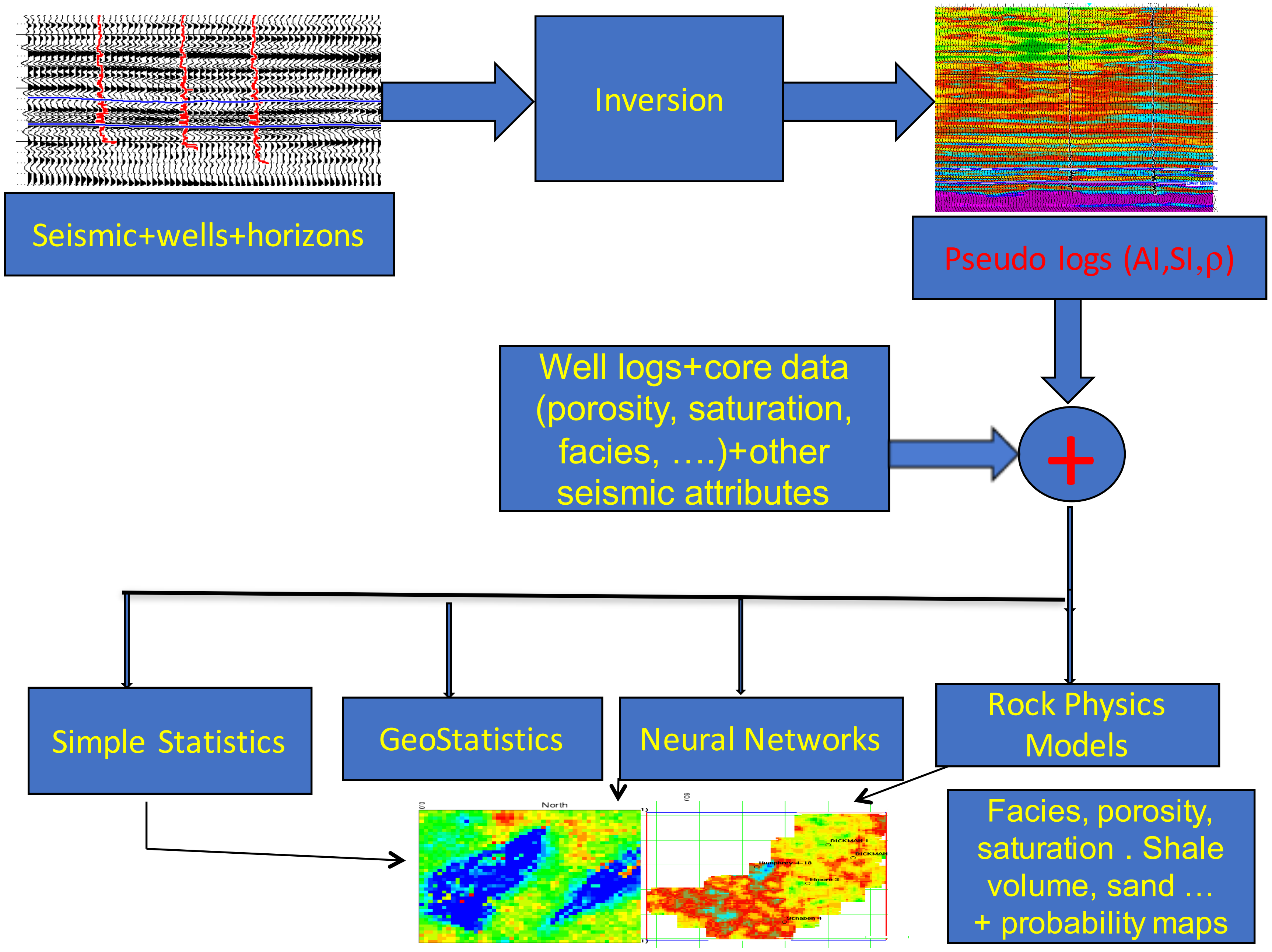
Machine Learning: Some applications

Mrinal K. Sen


Seismic+wells+horizons
Inversion
Pseudo logs (AI,SI,ρ)
Well logs+core data (porosity, saturation, facies, ….)+other seismic attributes
+
Simple Statistics
GeoStatistics
Neural Networks
Rock Physics Models
North
Facies, porosity, saturation . Shale volume, sand …
+ probability maps

Machine Learning

- Machine Learning is a branch of computer science
- It gives computers the ability to learn without being explicitly programmed. (Samuel 1959)
- Evolved from the study of pattern recognition and computer learning theory in artificial intelligence
- Machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Inversion and Machine Learning

Data Vector: $\mathbf{d} = [d_1\, d_2\, d_3 \ldots d_N]^T$

Model Vector $\mathbf{m} = [m_1 m_2\, m_3 \ldots m_M]^T$ Unknown!

Forward Modeling $\mathbf{d} = g(\mathbf{m})$

Linear $\mathbf{d} = \mathbf{G}\mathbf{m}$

In General $n \neq m$

Linear Problem – least squares solution $\mathbf{m}_{est} = \left[\mathbf{G}^T\mathbf{G}\right]^{-1}\mathbf{G}^T\mathbf{d}$,

Non-linear Problem – Local Optimization Global Optimization

Machine Learning

Forward Modeling operator is unknown!

Goal: Find an operator that can be applied to the data to estimate models

Find the operator by systematic examination of a series of observed data and their known answers. Learn from experience! **TRAINING**

$$\mathbf{m}_{est} = F\mathbf{d}$$

TRAINING

- **Supervised**
- **Unsupervised**

Supervised and Unsupervised Learning

- Machine Learning is a class of algorithms which is data-driven, i.e. unlike "normal" algorithms it is the data that "tells" what the "good answer" is.
- Example: a hypothetical non-machine learning algorithm for face detection in images would try to define what a face is (round skin-like-colored disk, with dark area where you expect the eyes etc).
- A machine learning algorithm would not have such coded definition, but would "learn-by-examples": you'll show several images of faces and not-faces and a good algorithm will eventually learn and be able to predict whether or not an unseen image is a face.
- This particular example of face detection is **supervised**, which means that your examples must be *labeled*, or explicitly say which ones are faces and which ones aren't.

Supervised and Unsupervised Learning

- In an **unsupervised** algorithm your examples are not *labeled*, i.e. you don't say anything. Of course, in such a case the algorithm itself cannot "invent" what a face is, but it can try to cluster the data into different groups, e.g. it can distinguish that faces are very different from landscapes, which are very different from horses.
- there are "intermediate" forms of supervision, i.e. **semi-supervised** and **active learning**. Technically, these are supervised methods in which there is some "smart" way to avoid a large number of labeled examples.
- In active learning, the algorithm itself decides which thing you should label (e.g. it can be pretty sure about a landscape and a horse, but it might ask you to confirm if a gorilla is indeed the picture of a face).
- In semi-supervised learning, there are two different algorithms which start with the labeled examples, and then "tell" each other the way they think about some large number of unlabeled data. From this "discussion" they learn.

History of Machine Learning

- Neural Networks (1960)
- Multi-layer Perceptions (1985)
- Restricted Boltzman Machines (1986)
- Support Vector Machine (1995)
- Deep Belief Networks – New interest in deep learning (2005)
- Deep Recurrent Neural Network (2009)
- Convolutional DBN (2010)
- Max Pooling CDBN (2011)

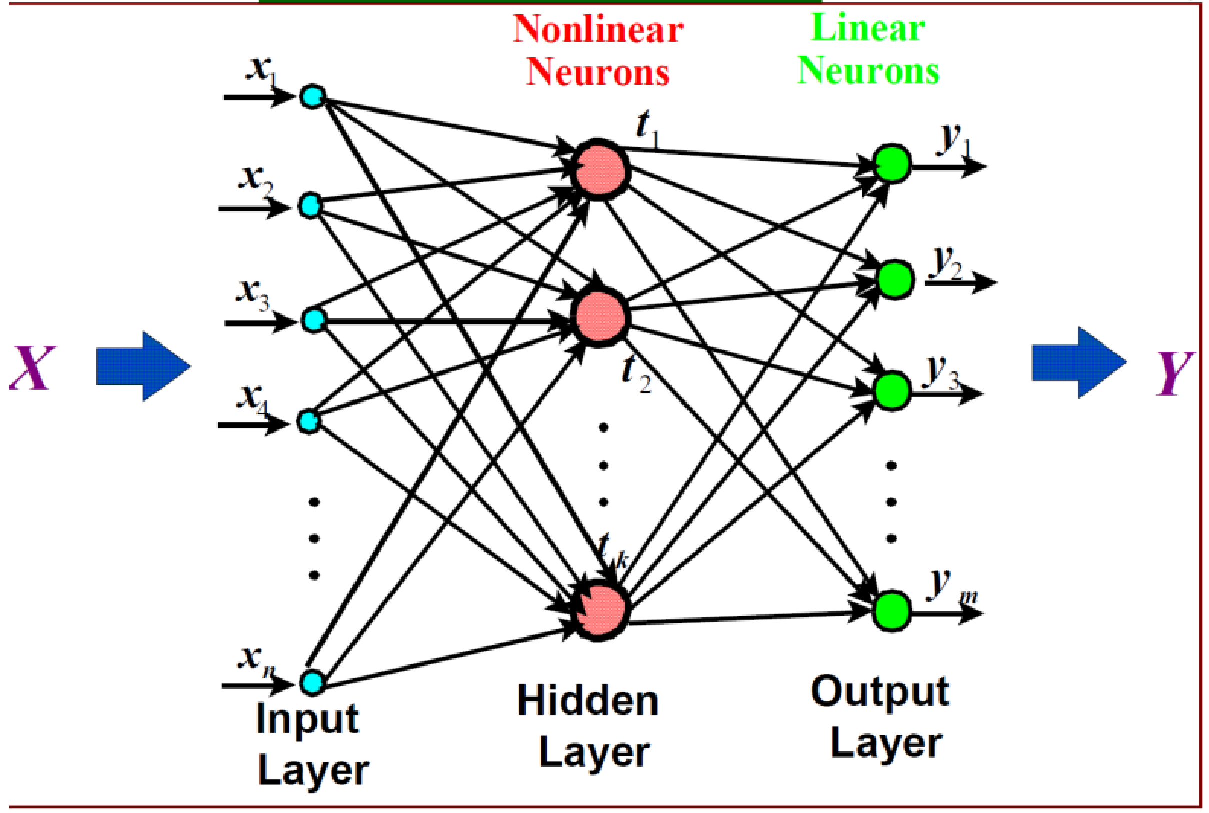
Mapping
Generalization of Function

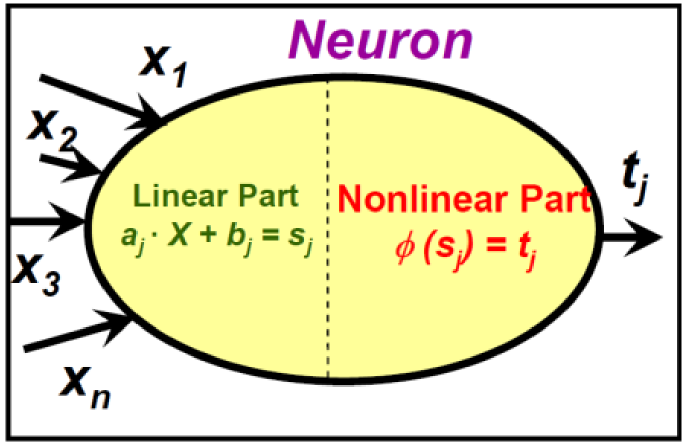
- **Mapping:** **A** *rule of correspondence established between vectors* **in vector spaces and that associates each vector** *X* **of a vector space** $\Re^n$ **with a vector** *Y* **in another vector space** $\Re^m$.

$$\left.\begin{aligned} &Y = F(X) \\ &X = \{x_1, x_2, \ldots, x_n\}, \in \Re^n \\ &Y = \{y_1, y_2, \ldots, y_m\}, \in \Re^m \end{aligned}\right\} \Rightarrow \begin{bmatrix} y_1 = f_1(x_1, x_2, \ldots, x_n) \\ y_2 = f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ y_m = f_m(x_1, x_2, \ldots, x_n) \end{bmatrix}$$

NN - Continuous Input to Output Mapping

Multilayer Perceptron: **Feed Forward, Fully Connected**





$$t_j = \phi(b_{j0} + \sum_{i=1}^{n} b_{ji} \cdot x_i) =$$

$$= \tanh(b_{j0} + \sum_{i=1}^{n} b_{ji} \cdot x_i)$$

Y = F

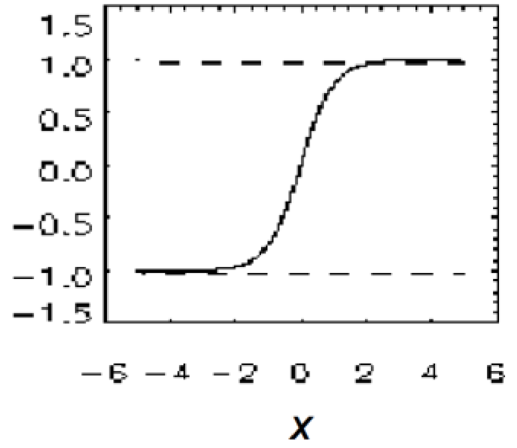
$Y = F_{NN}(X)$ $\Rightarrow$

Jacobian !

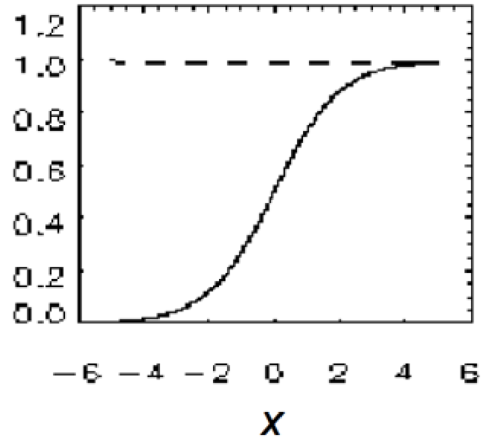
$$\begin{cases} y_q = a_{q0} + \sum_{j=1}^{k} a_{qj} \cdot t_j = a_{q0} + \sum_{j=1}^{k} a_{qj} \cdot \phi(\boxed{b_{j0}} + \sum_{i=1}^{n} \boxed{b_{ji}} \cdot x_i) = \\ = a_{q0} + \sum_{j=1}^{k} a_{qj} \cdot \tanh(b_{j0} + \sum_{i=1}^{n} b_{ji} \cdot x_i); \quad q = 1,2,\ldots,m \end{cases}$$

Some Popular Activation Functions

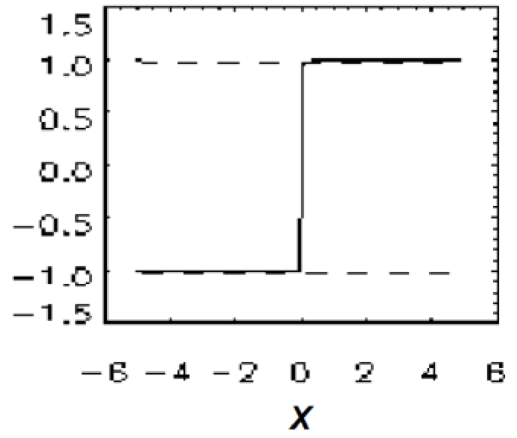
tanh(x)



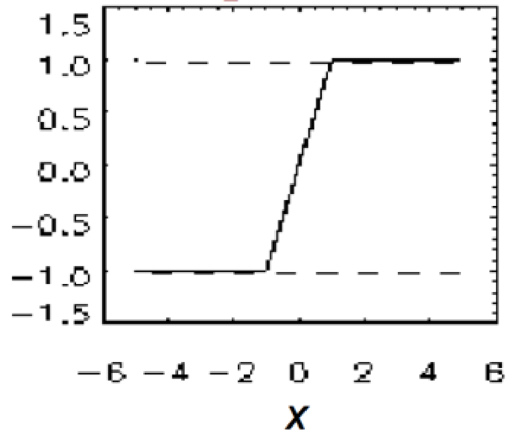
Sigmoid, $(1 + \exp(-x))^{-1}$



Hard Limiter



Ramp Function



Some references

- Calderón-Macías, C.*, **M. K. Sen**, and P. L. Stoffa, 2000, Neural Networks for Parameter Estimation in Geophysics, *Geophysical Prospecting*, 48, 21-47.
- Calderón-Macías, C.*, **M. K. Sen**, and P. L. Stoffa, 1998, Automatic NMO correction and velocity estimation by a feedforward neural network, *Geophysics,* 63(5), 1696-1707.
- Calderón-Macías, C.*, **M. K. Sen**, and P. L. Stoffa, 1997, Hopfield Neural Network and Mean Field Annealing in seismic deconvolution and multiple elimination, *Geophysics,* 62(3), 992-1002.
- Saraswat, P.*, V. Raj, **M. K. Sen**, and A. Naryanan, 2014, Multiattribute Seismic Analysis With Fractal Dimension and 2D and 3D Continuous Wavelet Transform, paper no. SPE-164417-MS, **http://dx.doi.org/10.2118/164417-PA**
- Saraswat, P.*, and **M. K. Sen**, 2012, Artificial neural systems based self organizing maps for seismic facies analysis, *Geophysics*, 77(4), O45-O53
- Srinivasan, S and **M. K. Sen,** 2009, Stochastic modeling of facies distribution in a carbonate reservoir in the Gulf of Mexico, *Geohorizons, December, 54-67*.
- Jin, L., **M. K. Sen**, and P. L. Stoffa, 2009, Fusion based classification method and its application, *Journal of Seismic Exploration*, 18(2), 103-117.
- Barone, A.*, and **M. K. Sen**, 2017, An Improved Classification Method that Combines Feature Selection with Nonlinear Bayesian Classification and Regression: A Case Study on Pore Fluid Prediction, SEG expanded abstracts.
- **Sen, M. K.**, and R. Biswas*, 2017, Trans-dimensional seismic inversion using the Hamiltonian Monte Carlo approach, Geophysics, 82(3), R119-R134.

EXAMPLE 1

Calderon-Macias and Sen

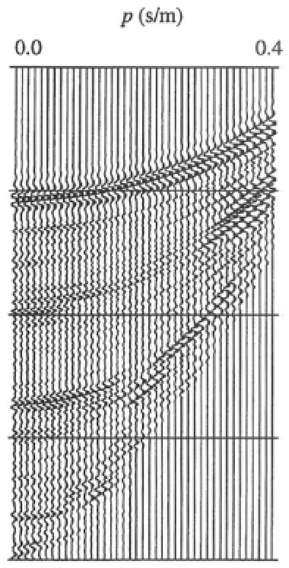
NMO Correction in τ-p domain

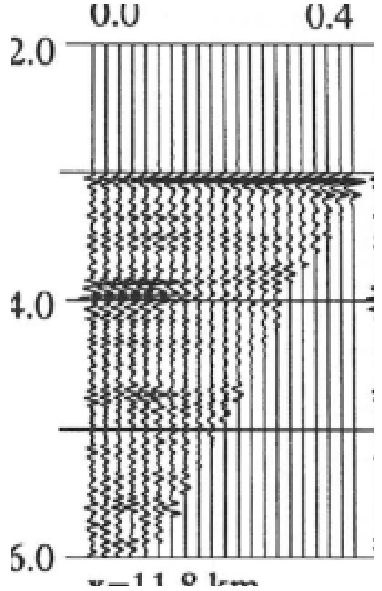
Moveout Equation

$$\Delta\tau_k(p) = \Delta\tau_k(0)\left(1 - p^2 v_k^2\right)^{1/2}$$

Measure of Goodness

$$h_1 = 1 - \frac{2\sum_{i=1}^{M}\sum_{k=1}^{N}|y_k - x_{ik}|}{\sum_{i=1}^{M}\sum_{k=1}^{N}|y_k + x_{ik}| + \sum_{i=1}^{M}\sum_{k=1}^{N}|y_k - x_{ik}|},$$





Steps

- The input data of the FNN are tau-p—transformed CMP gathers at selected control locations in a seismic line.
- The number of input elements of a single training example is given by the number of p-traces and the number of samples per trace *($L = M \times N$).*
- Network weights and sigmoid activation functions map the input data into interval velocities, the network outputs.
- Velocities can be described by a fixed number of layers of known two-way normal traveltime, by spline coefficients, or by some other smooth representation.
- The number of velocity layers or spline coefficients used to define the 1-D velocity-time functions gives the number of neurons S in the output layer of the network.
- The limits of the activation functions in the network output are given by the search limits of each output parameter.

- Weights are first initialized with random numbers between—1 and 1.
- Once a set of input vectors have been mapped by the FNN to network outputs, a velocity-time function per input vector, the data are NMO corrected and the error evaluated.

- Notice that the error is not obtained directly from the FNN output but from the NMO-corrected data.

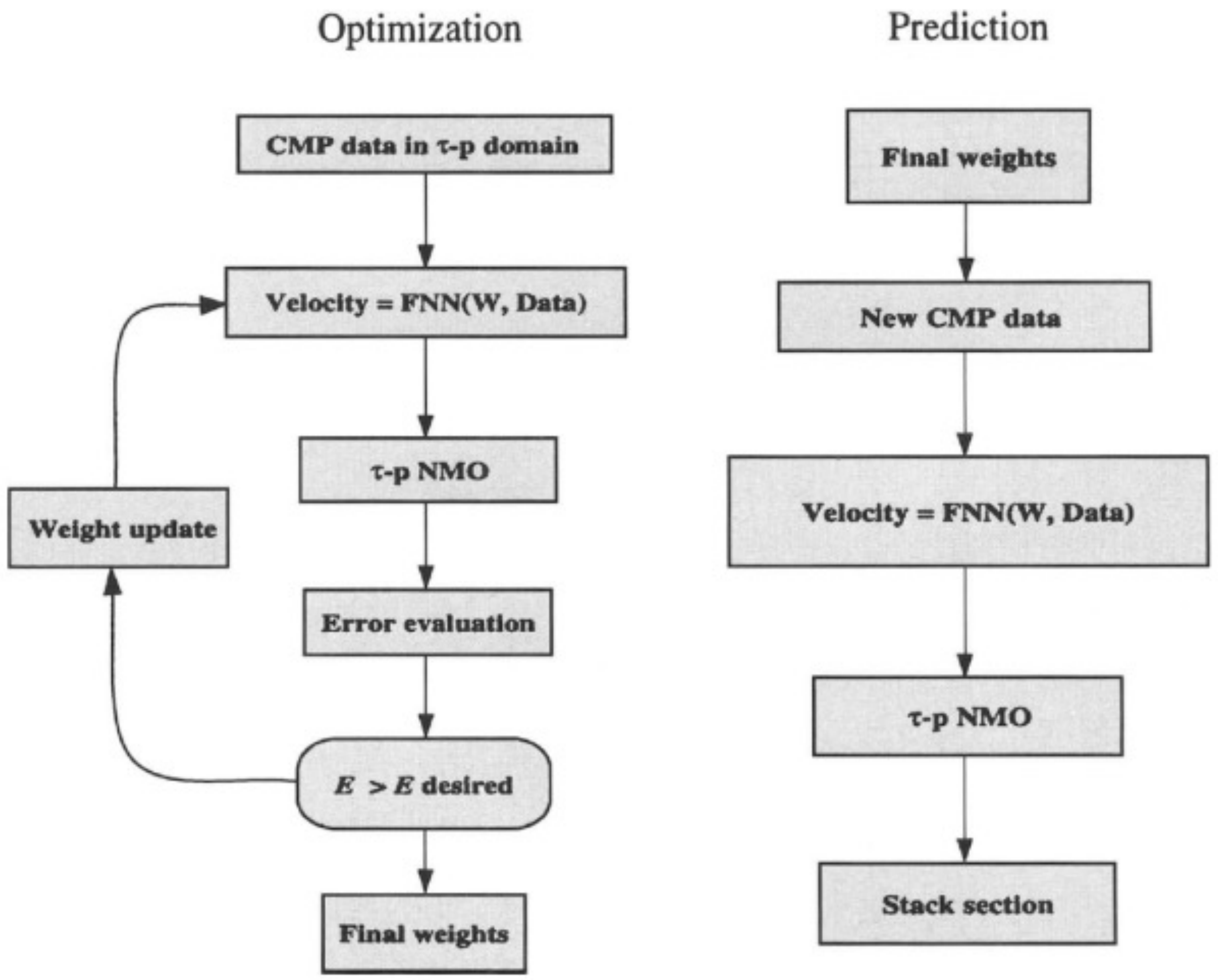


FIG. 2. Flow chart of the automatic NMO-correction method using FNNs.

Synthetic Example

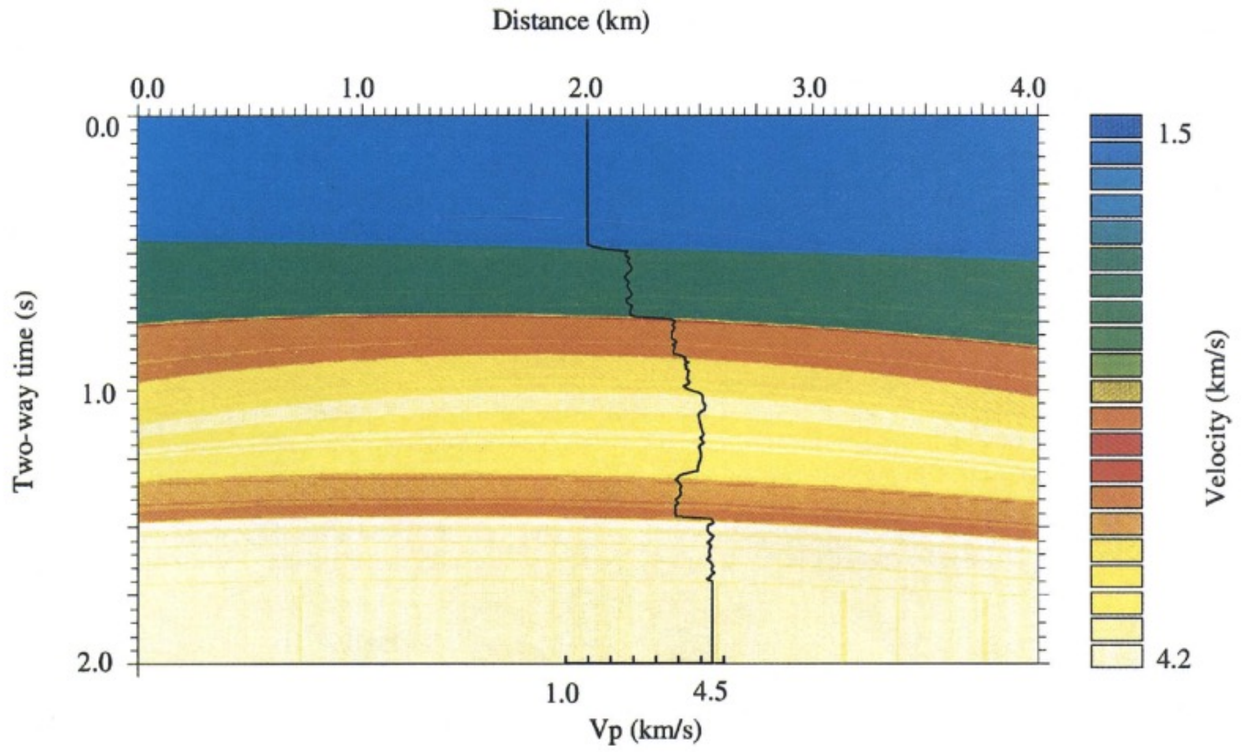


FIG. 3. Velocity model plotted in time. A velocity log is shown at the middle of the model.

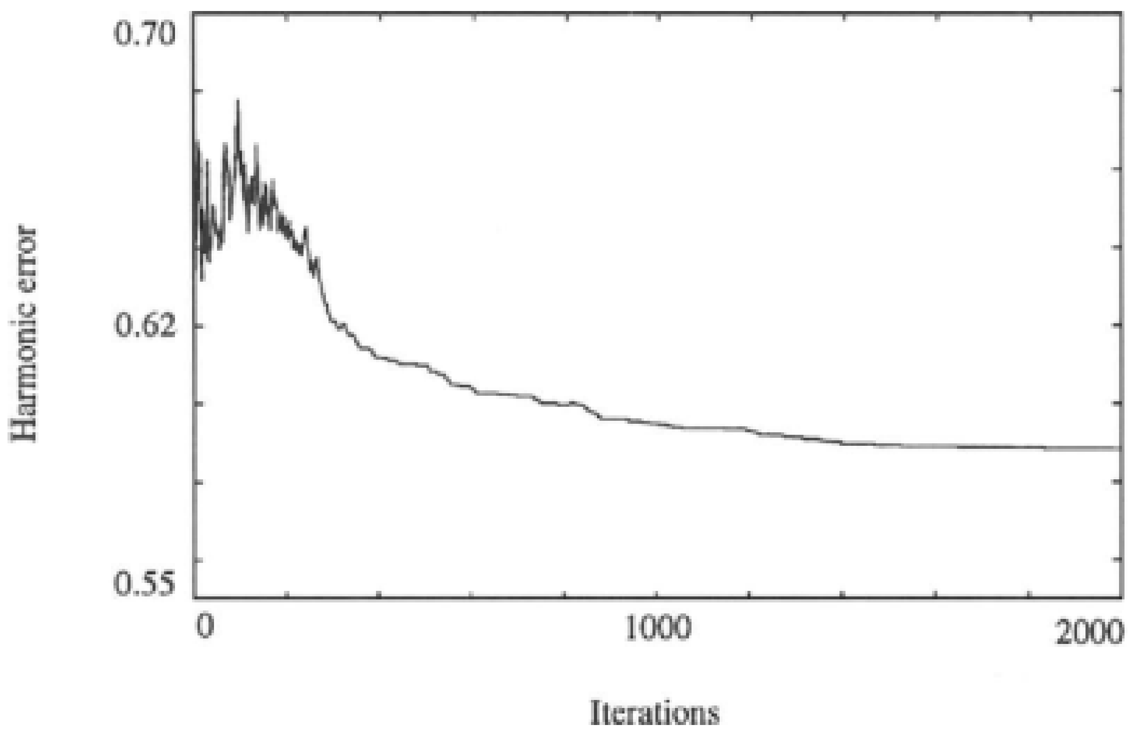


FIG. 5. Training history generated from NMO-correcting 11 CMP gathers.

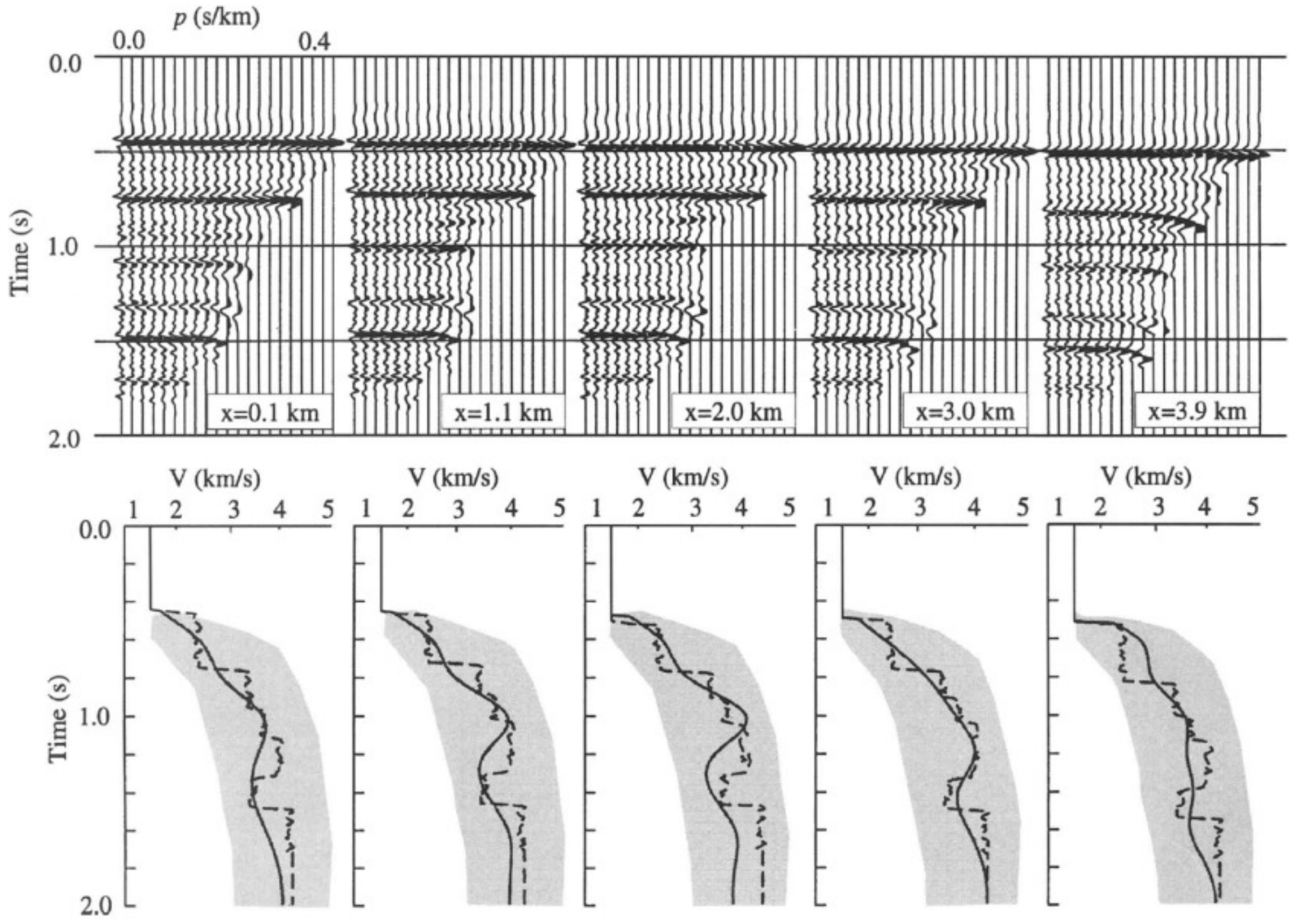


FIG. 6. NMO-correction results for five CMPs (top). True (dashed) and estimated (solid) velocities are shown for each of the CMP gathers (bottom); the shaded area corresponds to the velocity search range.

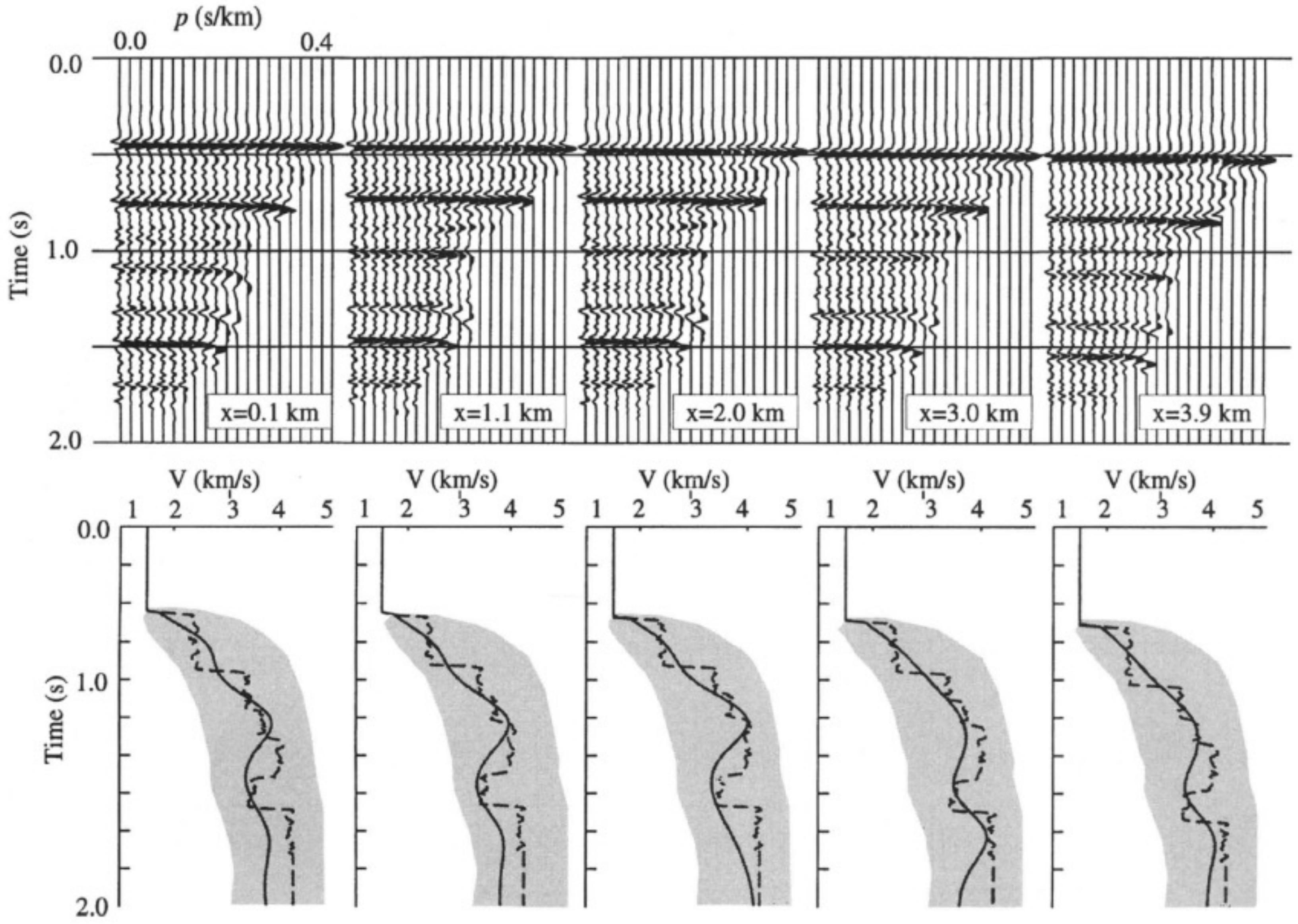


FIG. 7. Improved NMO-correction results after adding four examples for training from the edges of the model.

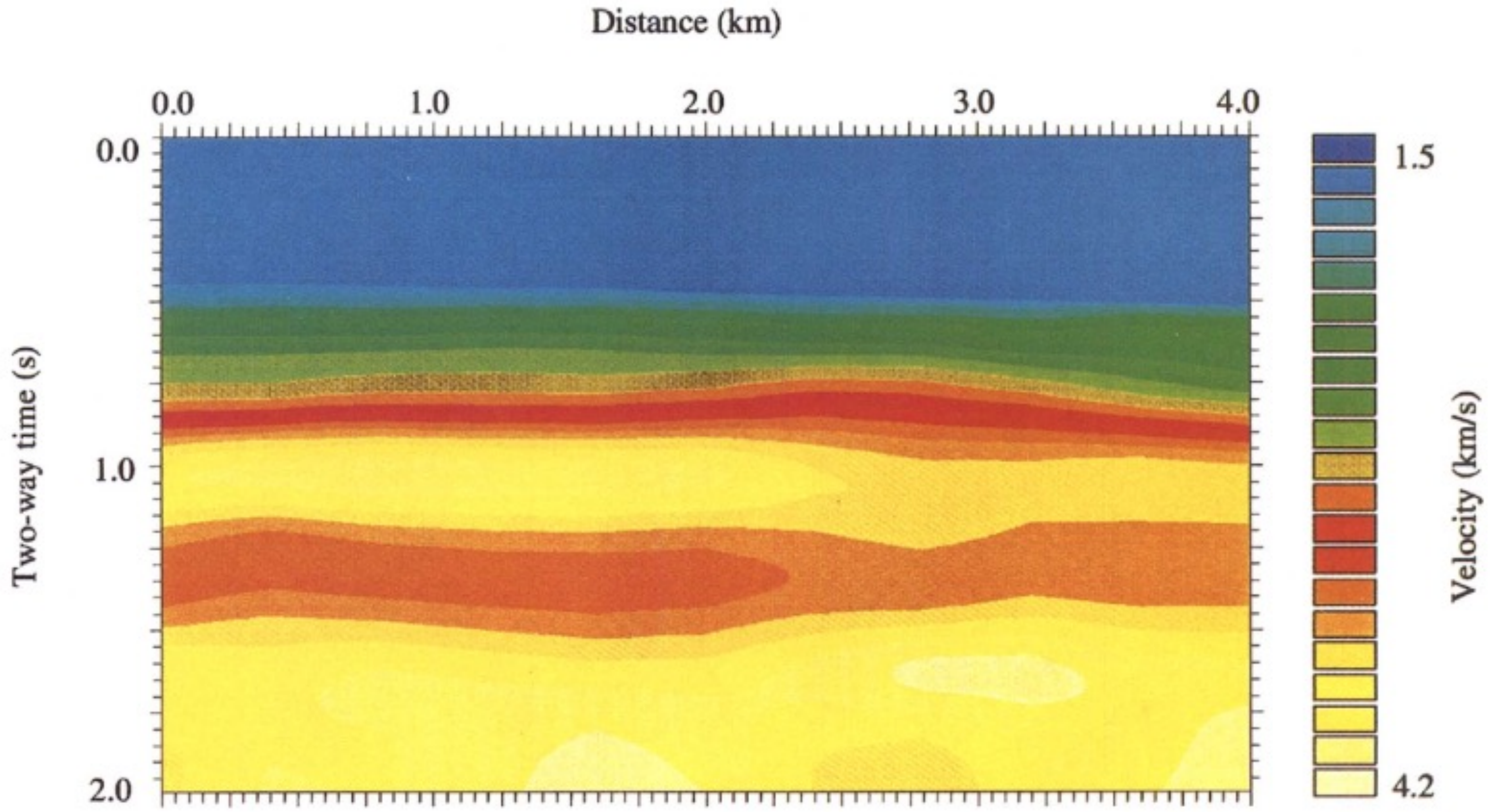


FIG. 8. Final velocity model from NMO correcting 100 CMPs using the trained FNN.

Field Data Example

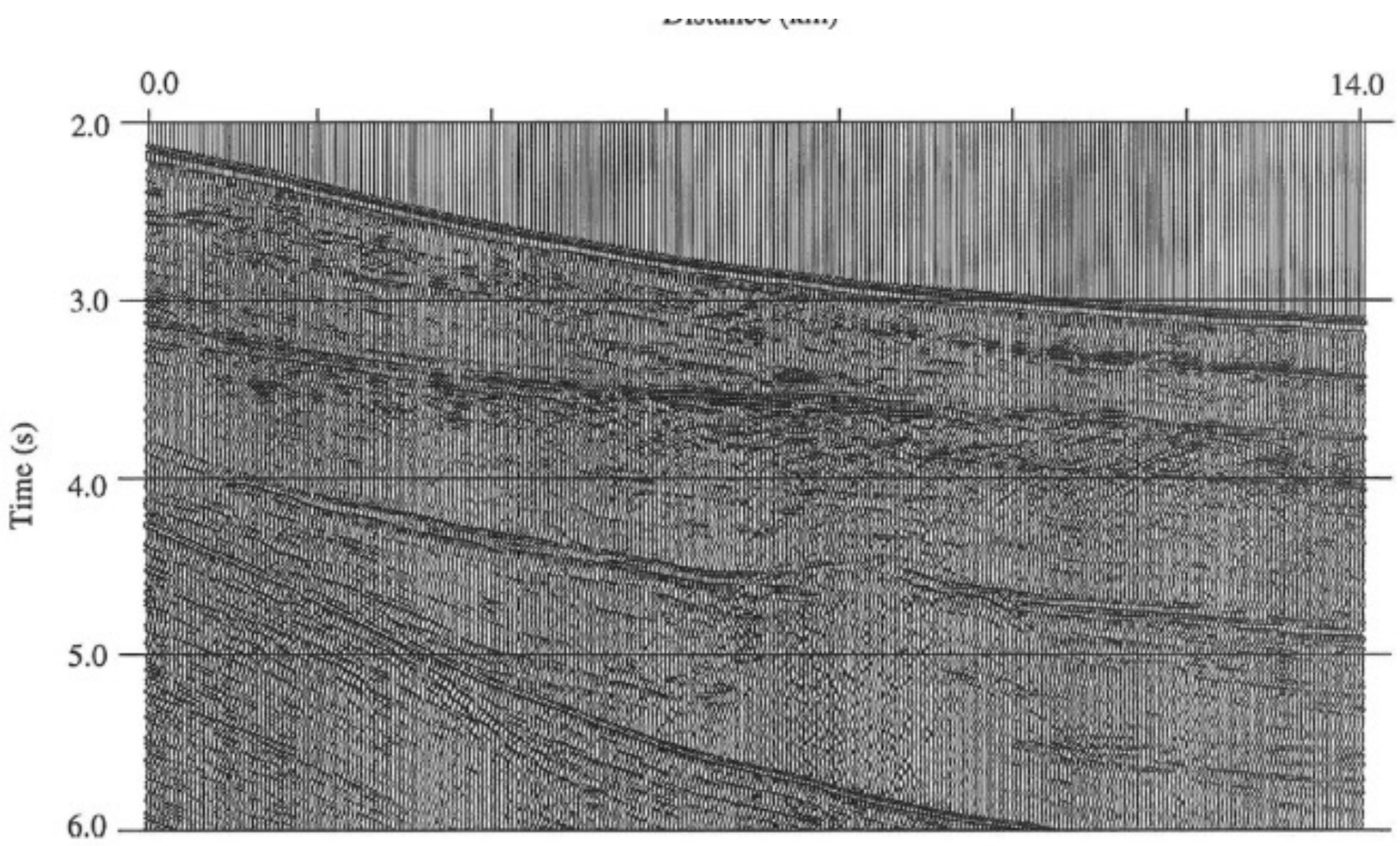


FIG. 11. Near trace section from the Carolina Trough data set.

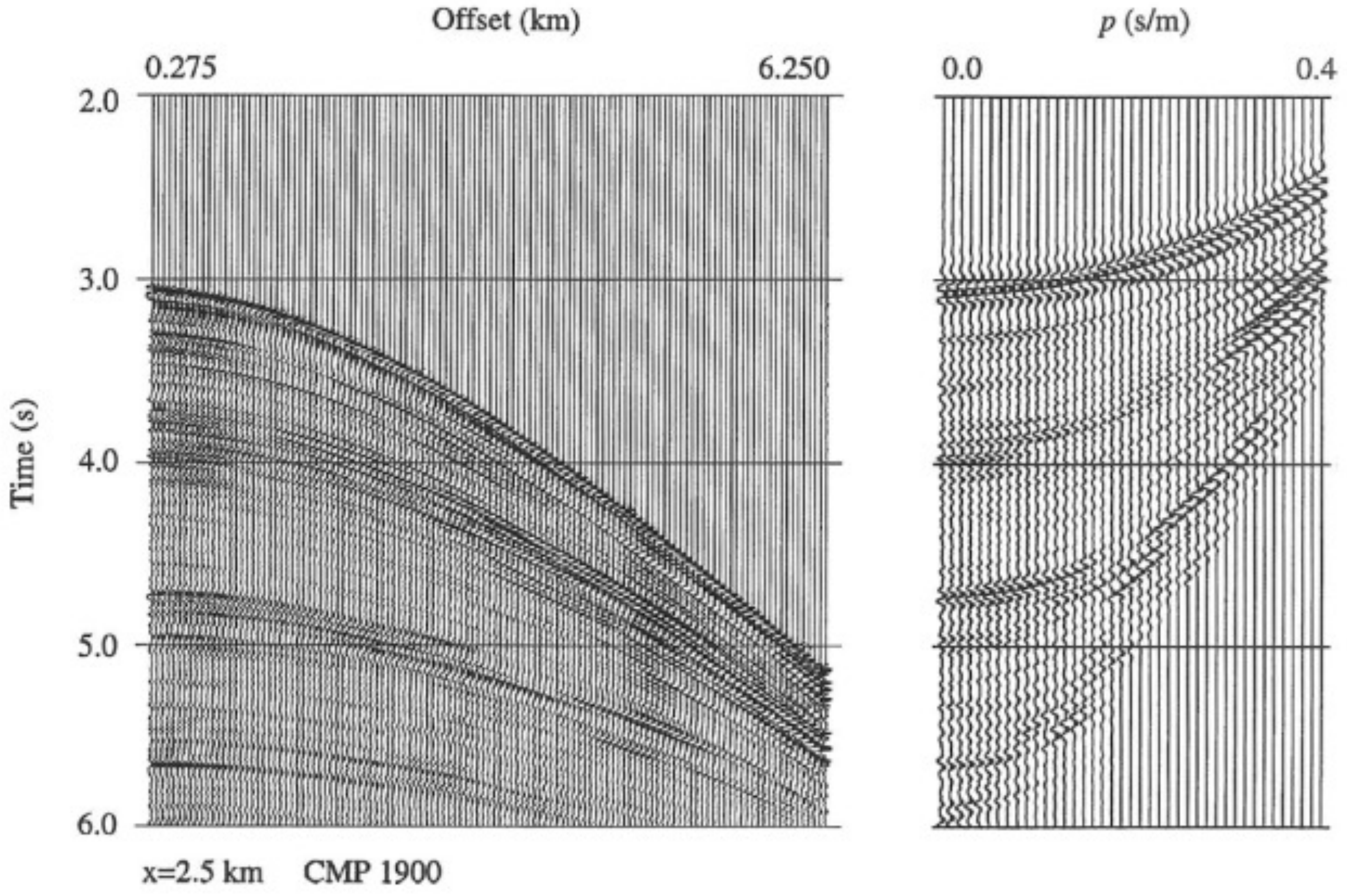


FIG. 12. Typical CMP gather from line BA6 (left) and its τ-p transform (right).

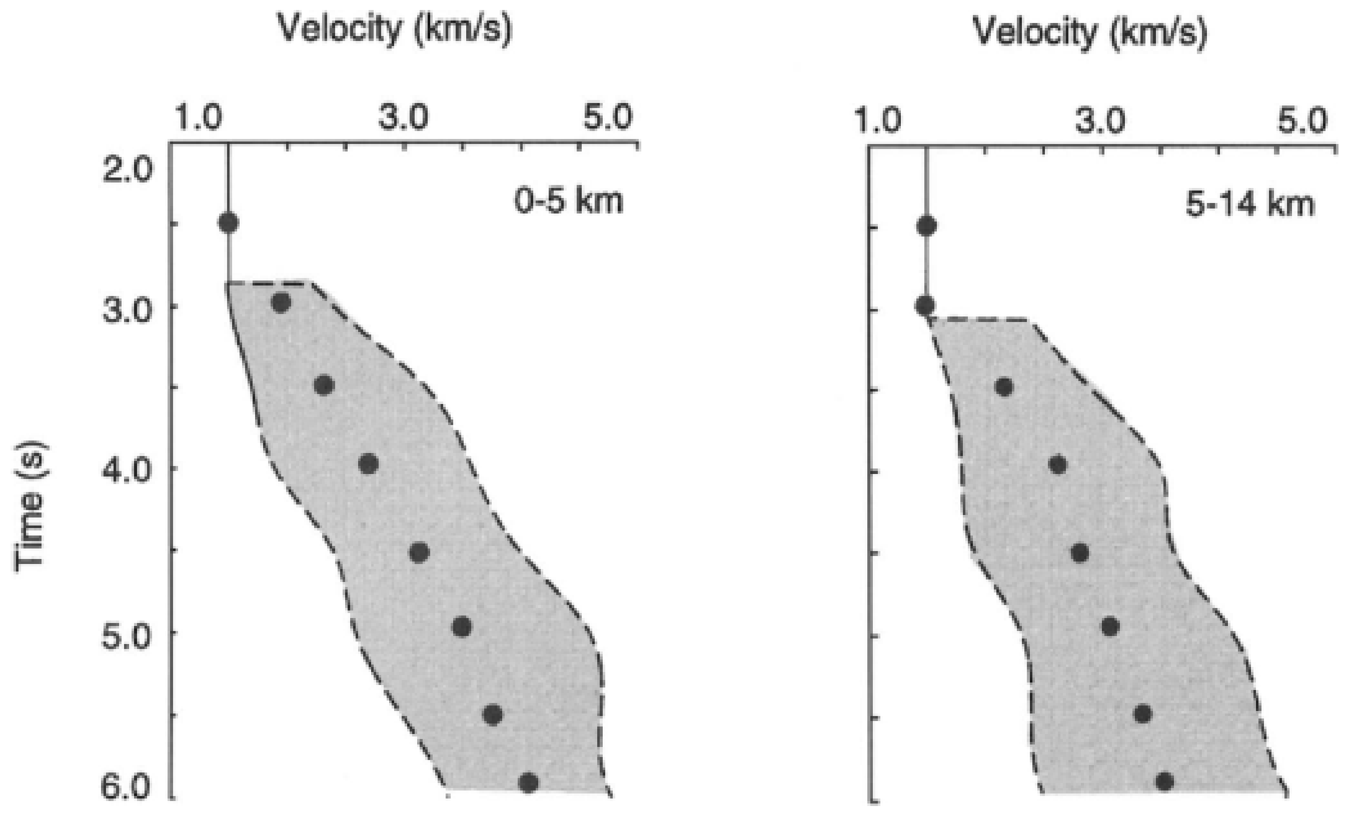


FIG. 13. Velocity search limits used for automatic NMO correction for the fist FNN (left) and second and third FNNs (right). Circles indicate the location of the spline nodes.

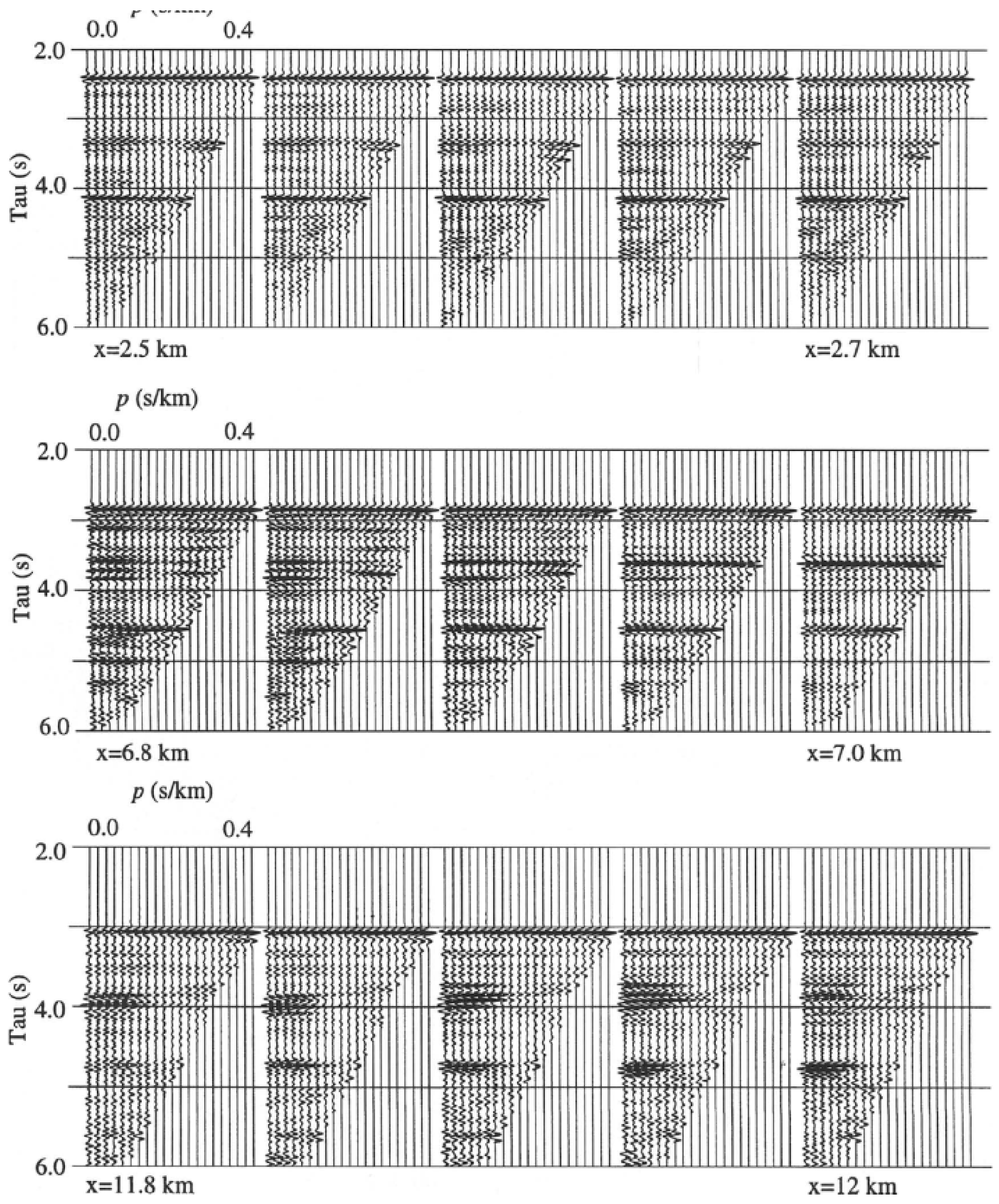


FIG. 14. NMO-correction interpolation results obtained from training three FNNs. The separation distance between CMP gathers in each group is 0.05 km. The CMP gathers at the extremes of each group correspond to control CMP gathers used for training.

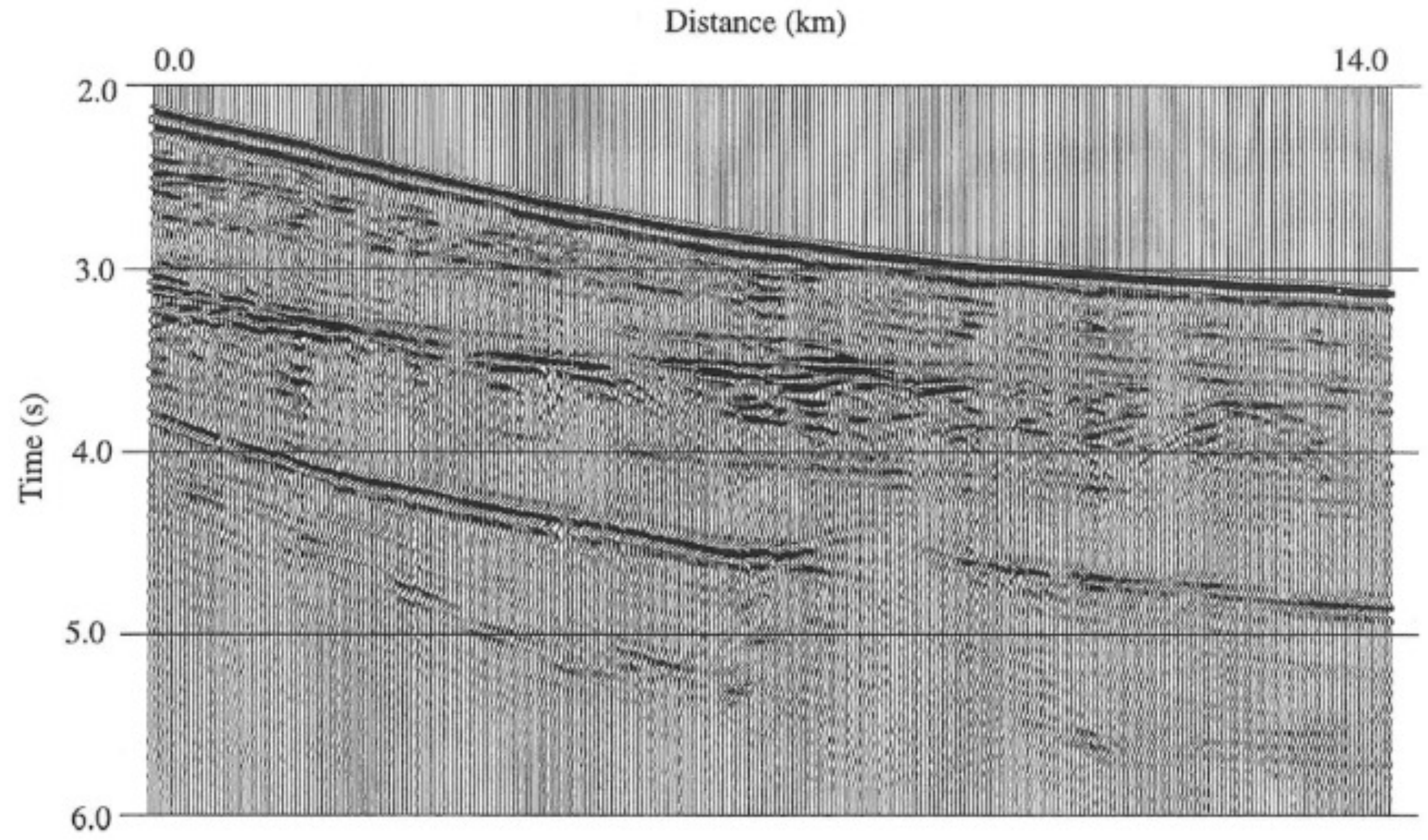
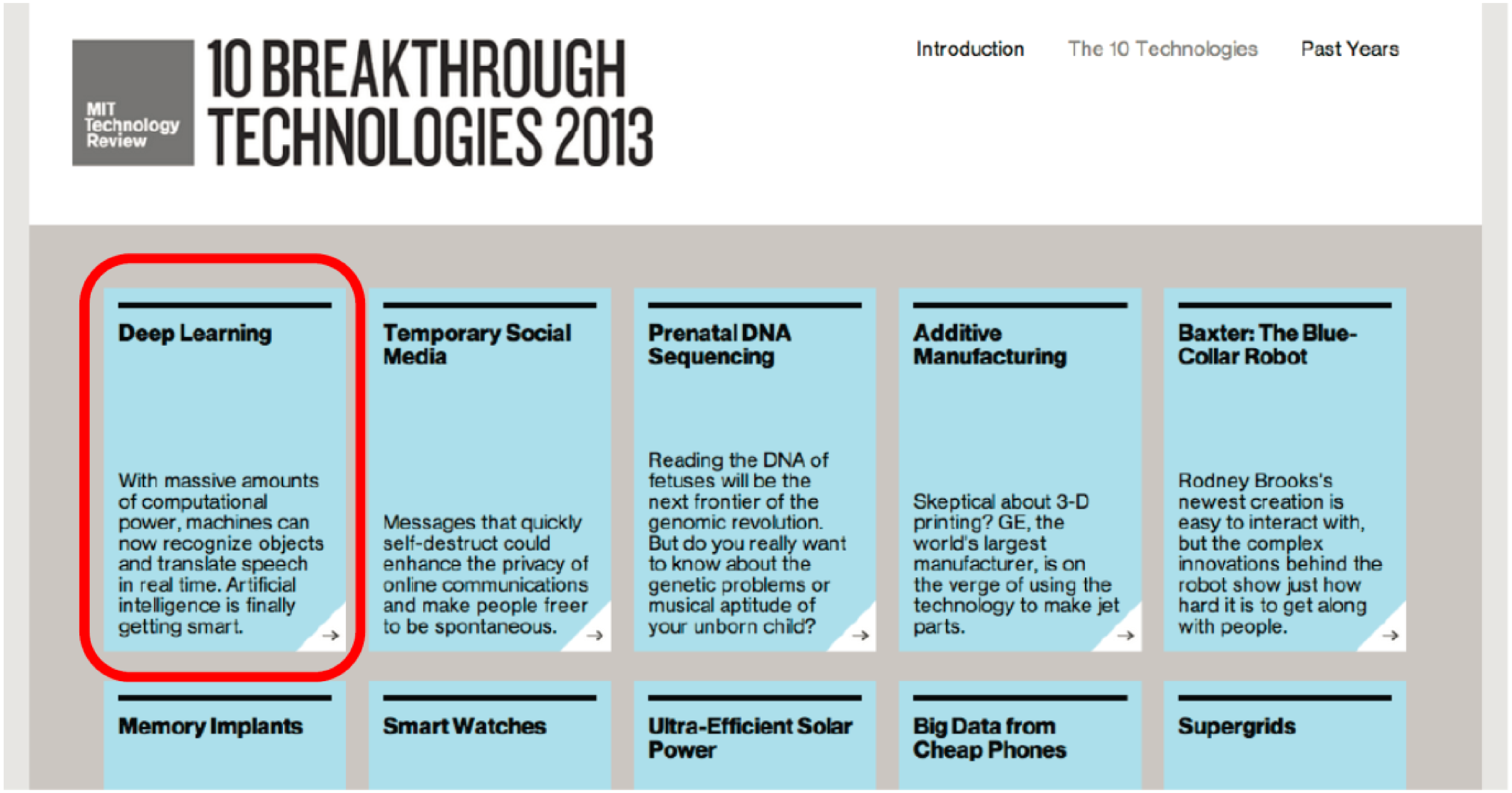


FIG. 15. Final stack section obtained after NMO-correcting CMP gathers with the trained networks as velocity interpolators. Fifty-six CMP gathers were used for training.



MIT Technology Review

10 BREAKTHROUGH TECHNOLOGIES 2013

Introduction The 10 Technologies Past Years

Deep Learning

With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.

Temporary Social Media

Messages that quickly self-destruct could enhance the privacy of online communications and make people freer to be spontaneous.

Prenatal DNA Sequencing

Reading the DNA of fetuses will be the next frontier of the genomic revolution. But do you really want to know about the genetic problems or musical aptitude of your unborn child?

Additive Manufacturing

Skeptical about 3-D printing? GE, the world's largest manufacturer, is on the verge of using the technology to make jet parts.

Baxter: The Blue-Collar Robot

Rodney Brooks's newest creation is easy to interact with, but the complex innovations behind the robot show just how hard it is to get along with people.

Memory Implants

Smart Watches

Ultra-Efficient Solar Power

Big Data from Cheap Phones

Supergrids

MIT Technology Review, April 23rd, 2013

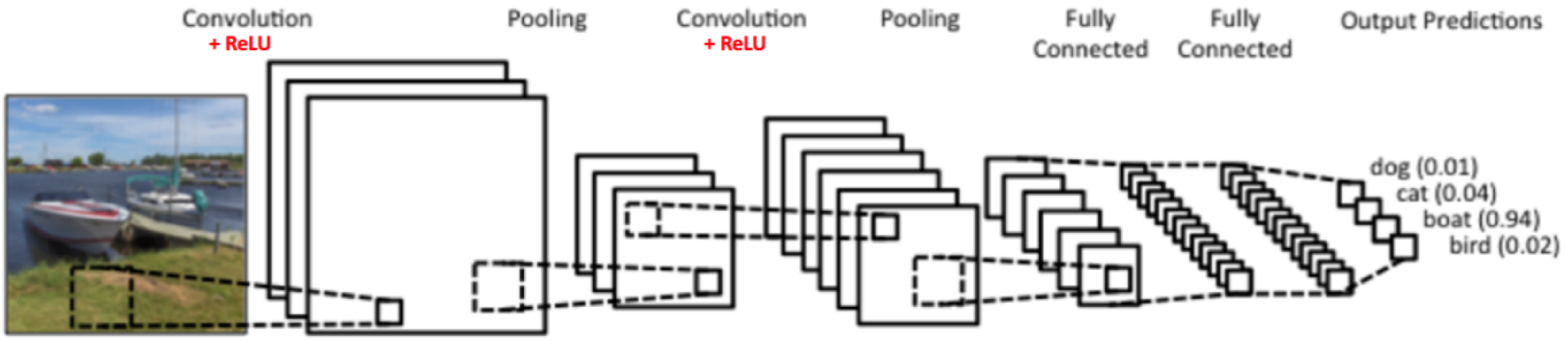
CNN

- CNNs are simply neural networks that use convolution in place of general matrix multiplication in at least one of the layers!

- In a traditional NN, every output unit interacts with every input unit.

- CNNs typically have sparse interactions accomplished by making the kernel smaller than input. Storage of fewer parameters and fewer operations!

Basic Architecture

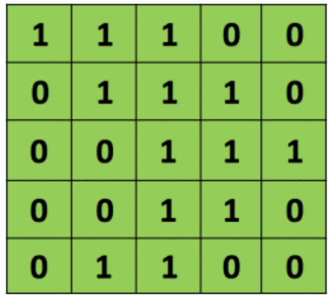


There are four main operations in the ConvNet shown in **Figure** above:

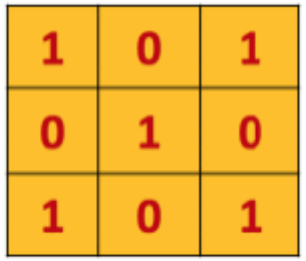
- Convolution
- Non Linearity (ReLU) [Rectified Linear Unit]
- Pooling or Sub Sampling
- Classification (Fully Connected Layer)

Convolution

- ConvNets derive their name from the “convolution” operator.
- The primary purpose of Convolution in case of a ConvNet is to extract features from the input image.
- Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.
- We will not go into the mathematical details of Convolution here, but will try to understand how it works over images.

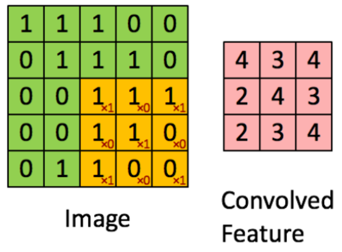


1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0



1	0	1
0	1	0
1	0	1

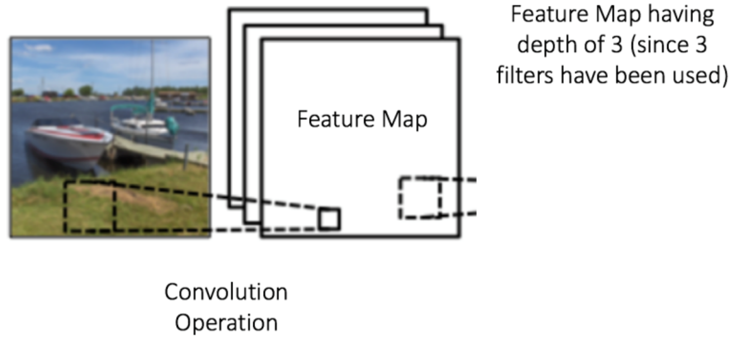
Filter or Kernel



Feature map

Terms

Depth: Depth corresponds to the number of filters we use for the convolution operation.


Feature Map having depth of 3 (since 3 filters have been used)
Feature Map
Convolution Operation

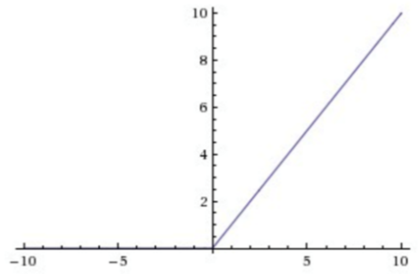
Stride: Stride is the number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time.

Zero-padding: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. A nice feature of zero padding is that it allows us to control the size of the feature maps.

Relu

ReLU stands for Rectified Linear Unit and is a non-linear operation. Its output is given by:

Output = Max(zero, Input)



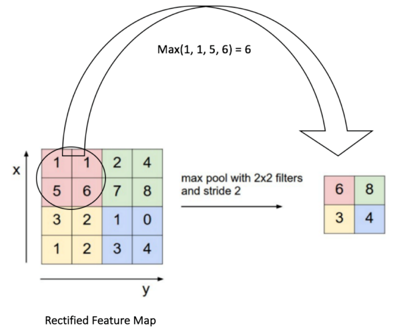
ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero.

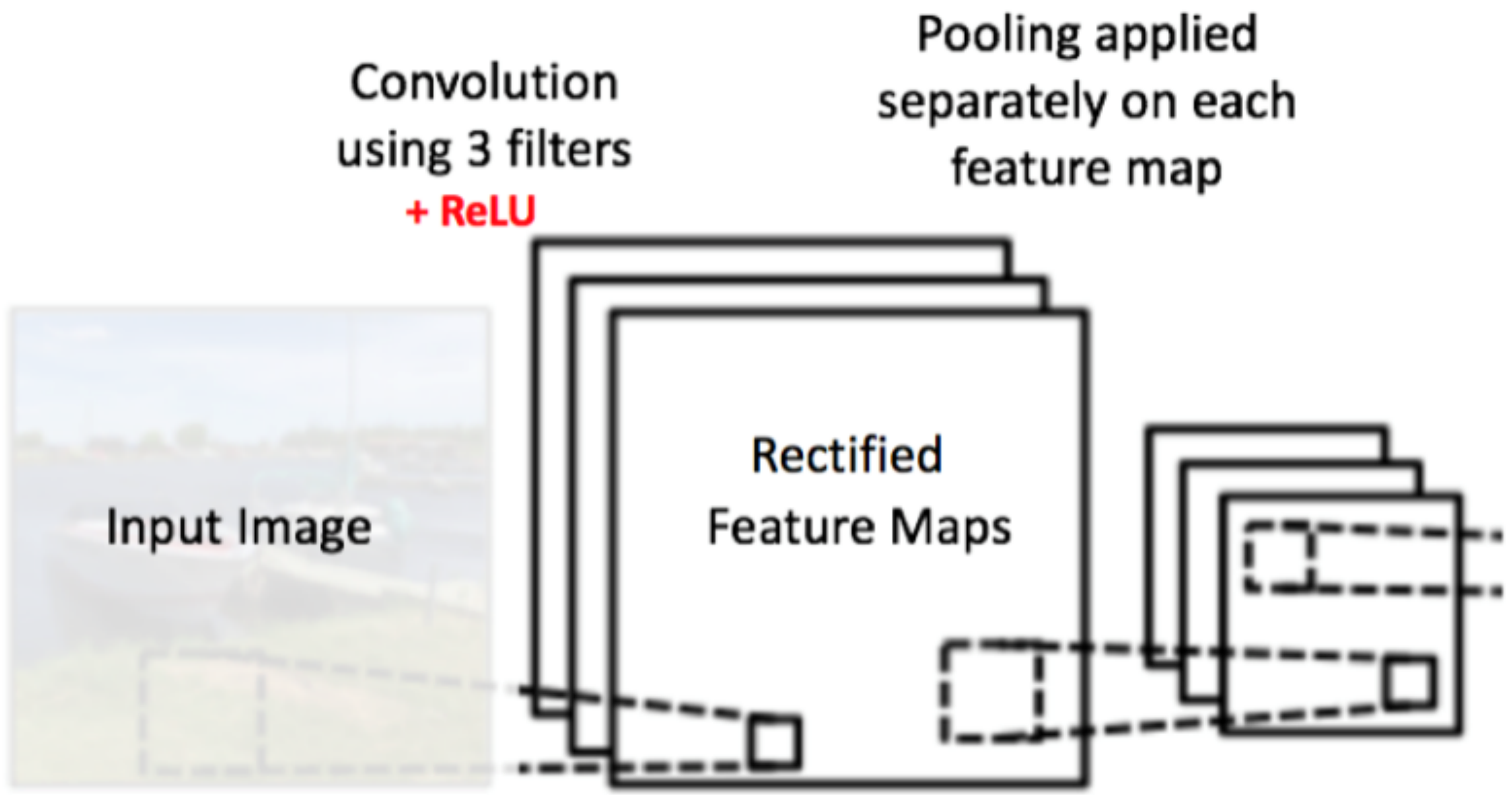
The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

Other non linear functions such as **tanh** or **sigmoid** can also be used instead of ReLU, but ReLU has been found to perform better in most situations.

Pooling

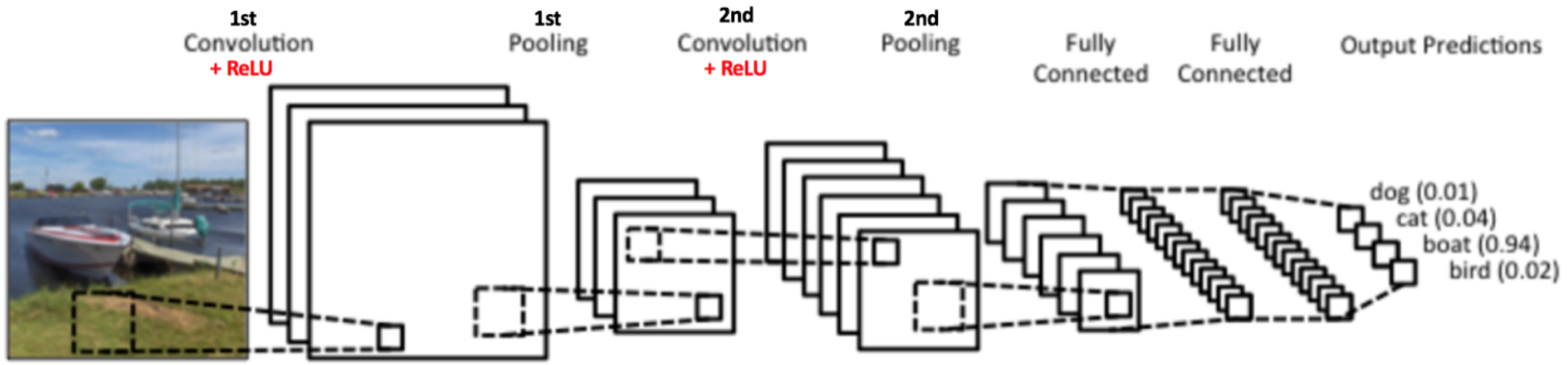
- Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.
- In case of Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better.




Convolution
using 3 filters
+ ReLU
Pooling applied
separately on each
feature map
Input Image
Rectified
Feature Maps

- The function of Pooling is to progressively reduce the spatial size of the input representation.
- In particular, pooling makes the input representations (feature dimension) smaller and more manageable
- reduces the number of parameters and computations in the network, therefore, controlling overfitting
- makes the network invariant to small transformations, distortions and translations in the input image (a small distortion in input will not change the output of Pooling – since we take the maximum / average value in a local neighborhood).
- helps us arrive at an almost scale invariant representation of our image (the exact term is “equivariant”). This is very powerful since we can detect objects in an image no matter where they are located.

Story so far



the 2nd Convolution layer performs convolution on the output of the first Pooling Layer using six filters to produce a total of six feature maps.

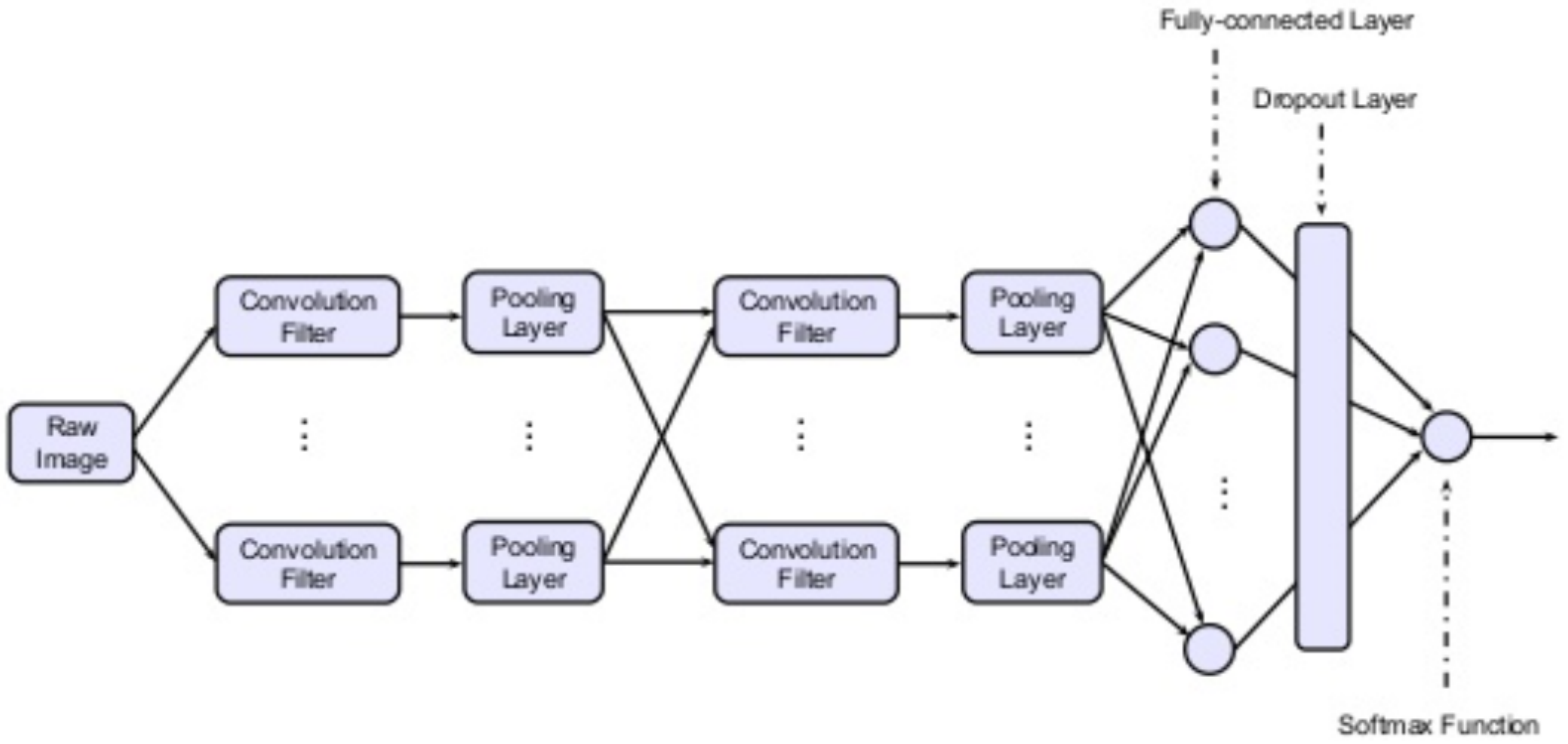
ReLU is then applied individually on all of these six feature maps.

We then perform Max Pooling operation separately on each of the six rectified feature maps.

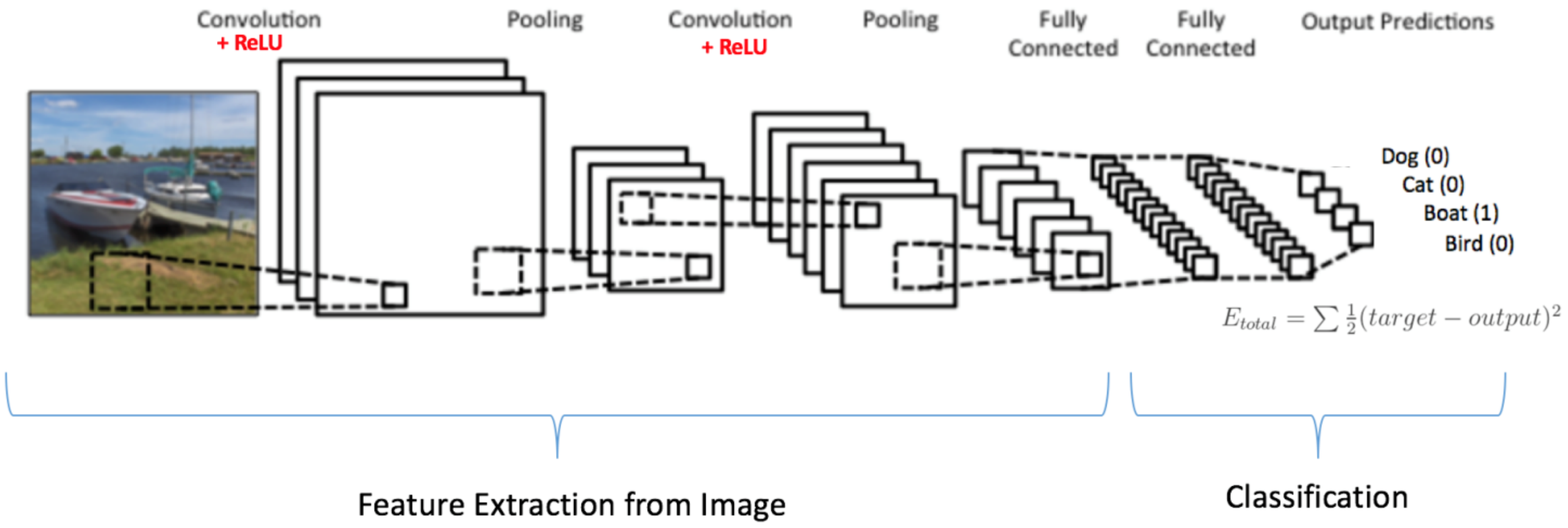
The output of the 2nd Pooling Layer acts as an input to the Fully Connected Layer,

Fully Connected Layer

- The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer (other classifiers like SVM can also be used, but will stick to softmax in this post). The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer.
- The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.
- The sum of output probabilities from the Fully Connected Layer is 1. This is ensured by using the Softmax as the activation function in the output layer of the Fully Connected Layer. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one.

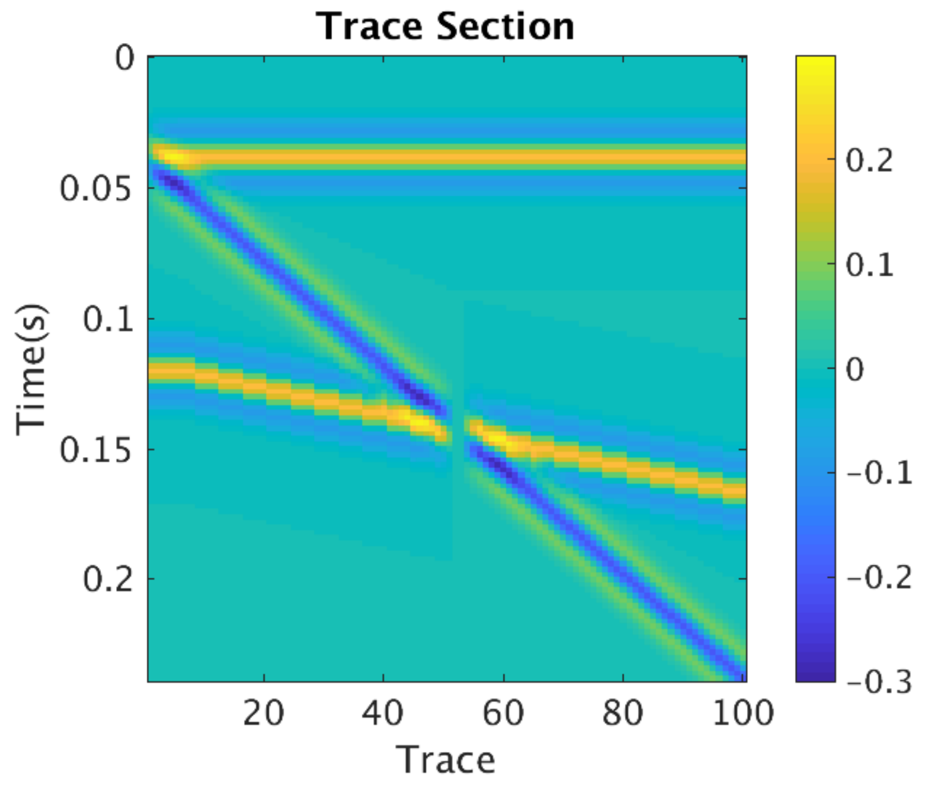

Fully-connected Layer
Dropout Layer
Raw Image
Convolution Filter
Pooling Layer
Convolution Filter
Pooling Layer
Convolution Filter
Pooling Layer
Convolution Filter
Pooling Layer
Softmax Function

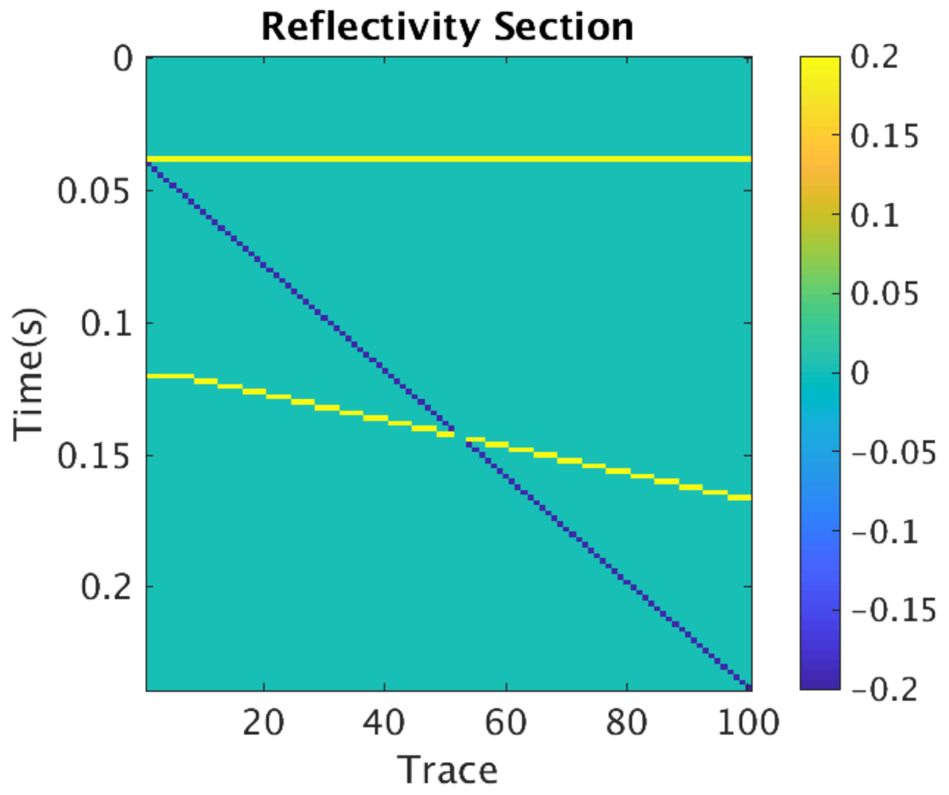
Training


Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
Fully Connected
Fully Connected
Output Predictions
Dog (0)
Cat (0)
Boat (1)
Bird (0)
$E_{total} = \sum \frac{1}{2}(target - output)^2$
Feature Extraction from Image
Classification

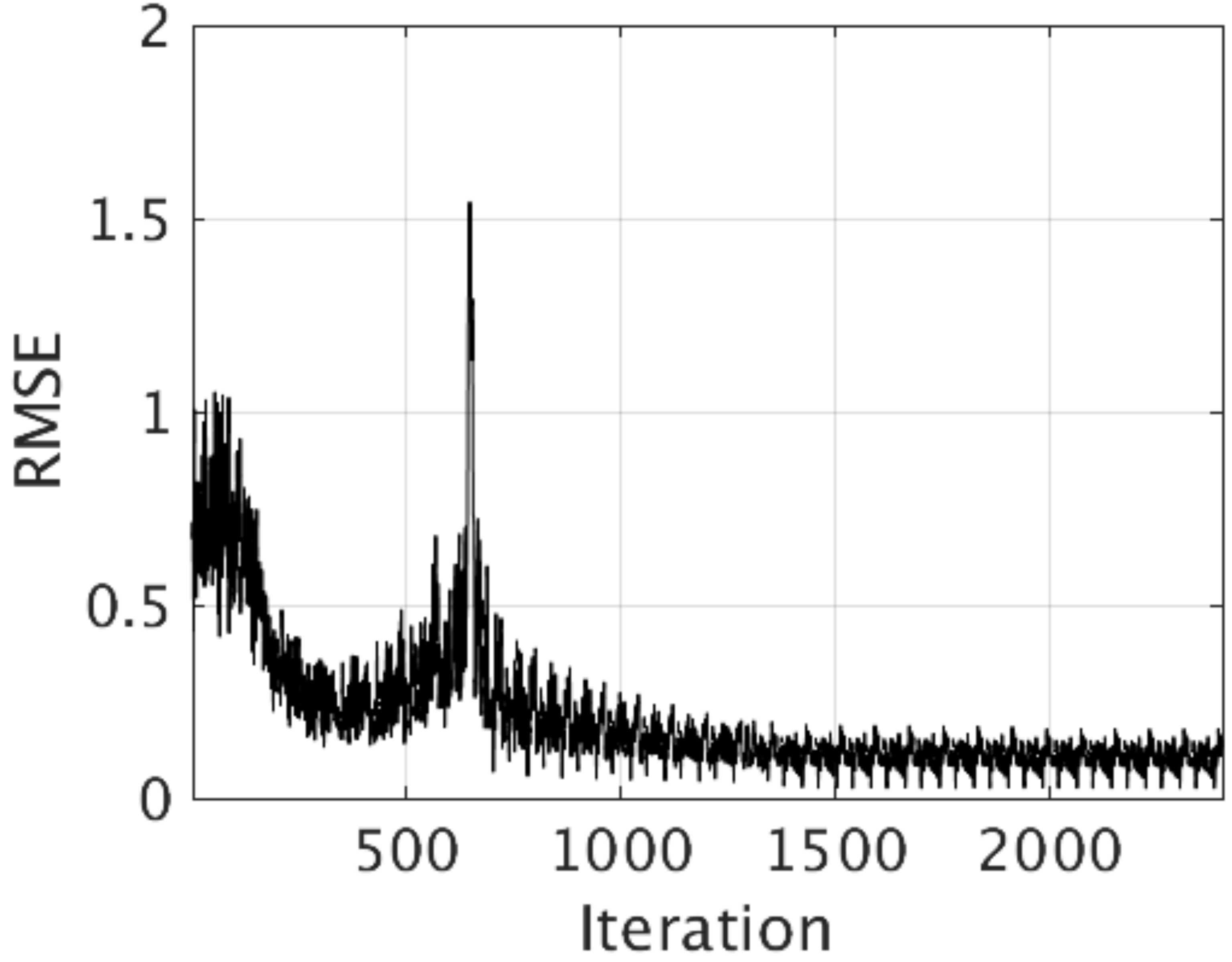
Application

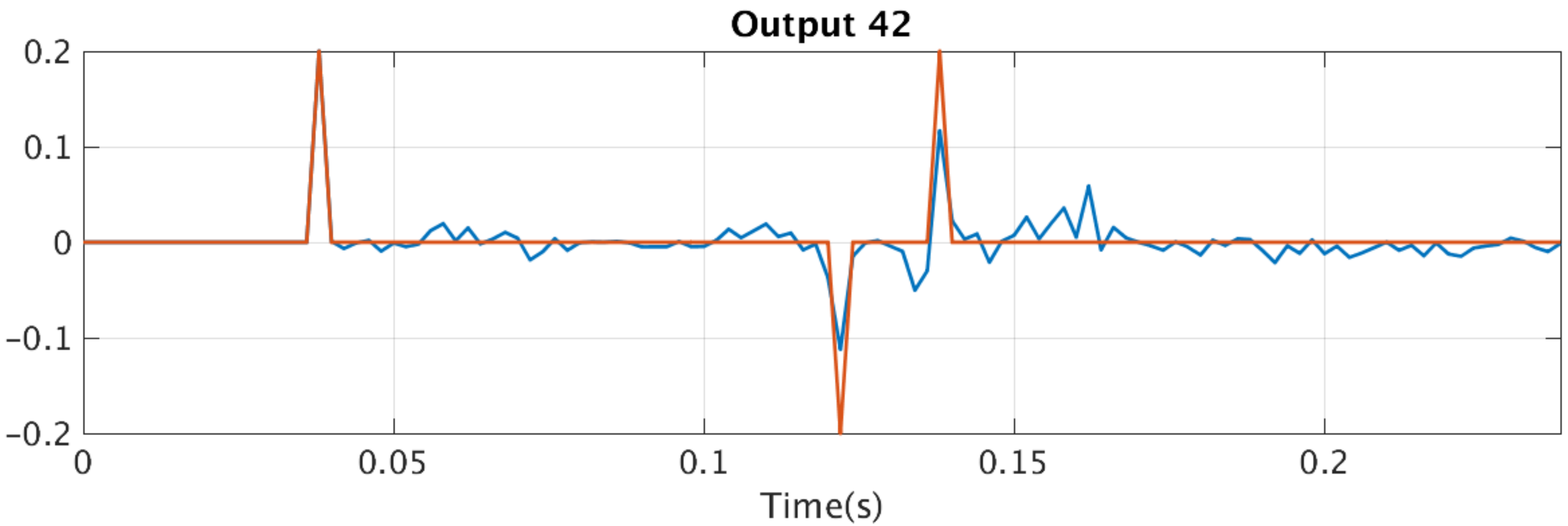
- 1D Seismic Deconvolution problem

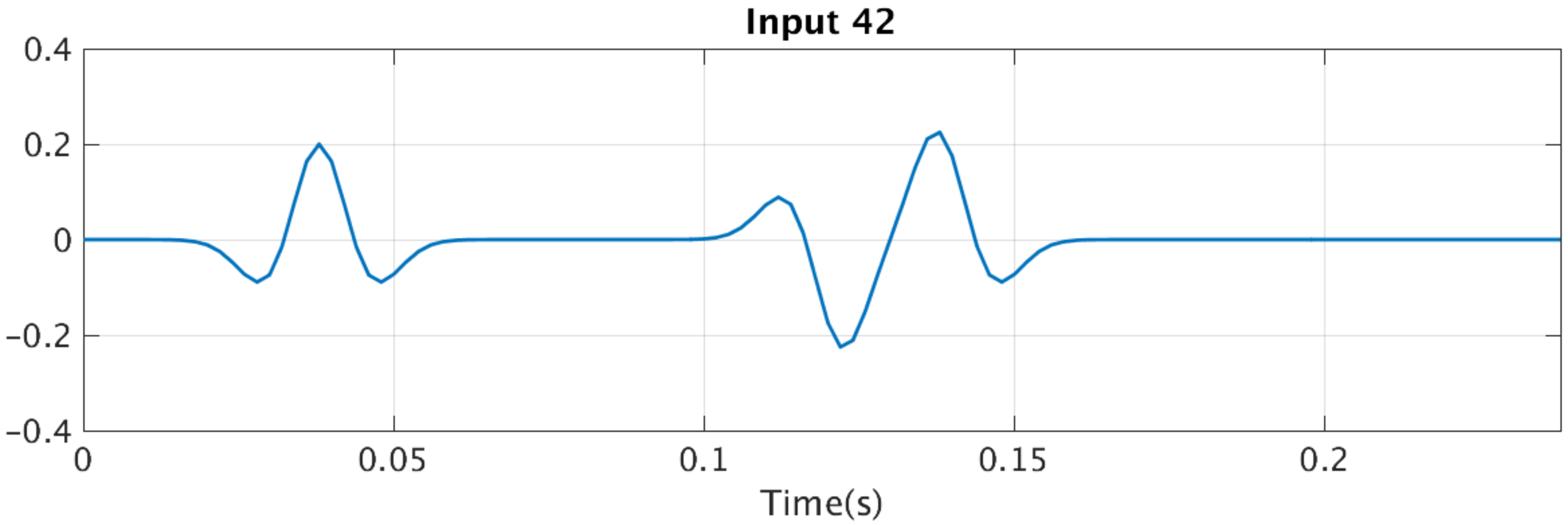




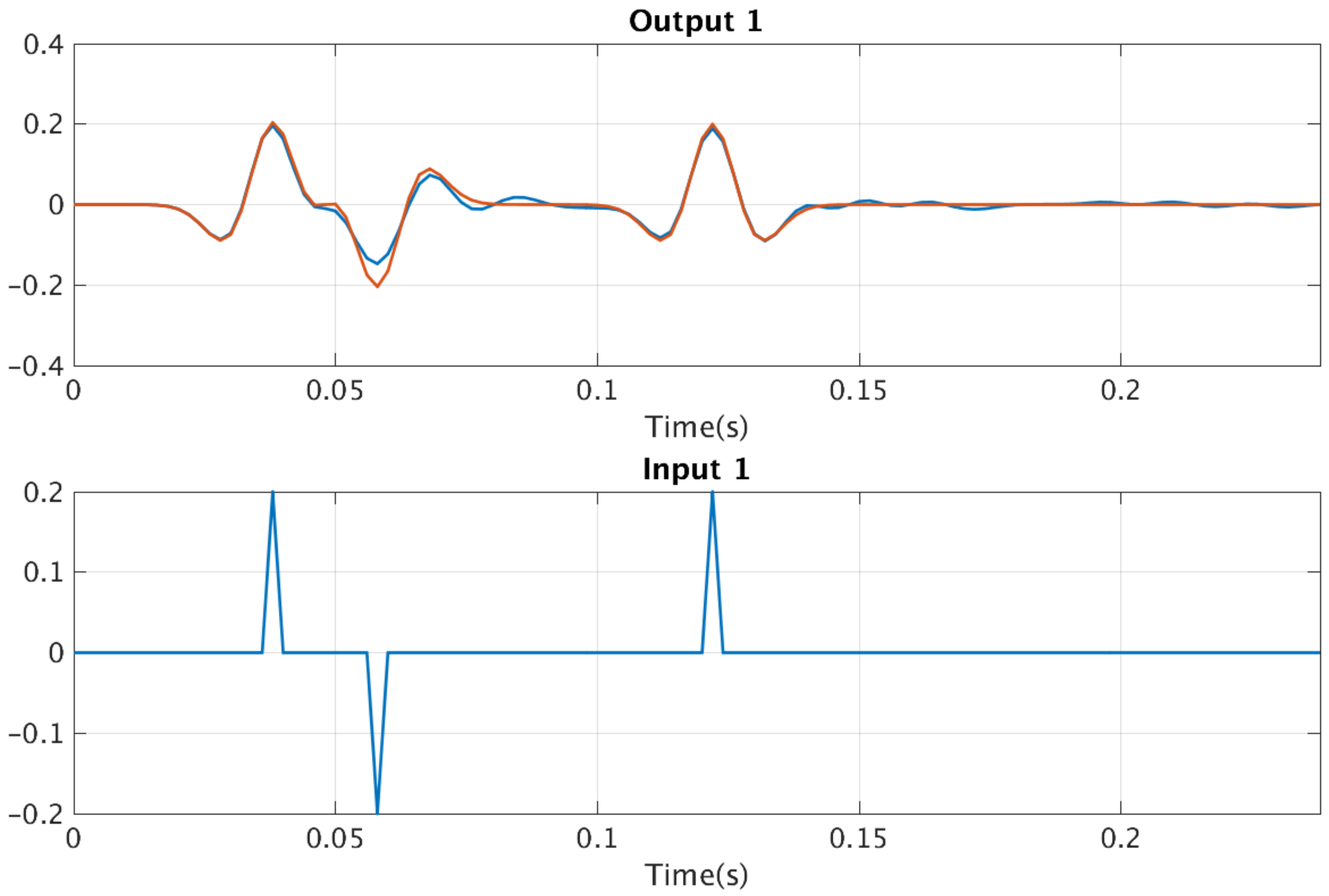
FNN


2
1.5
1
0.5
0
RMSE
500
1000
1500
2000
Iteration

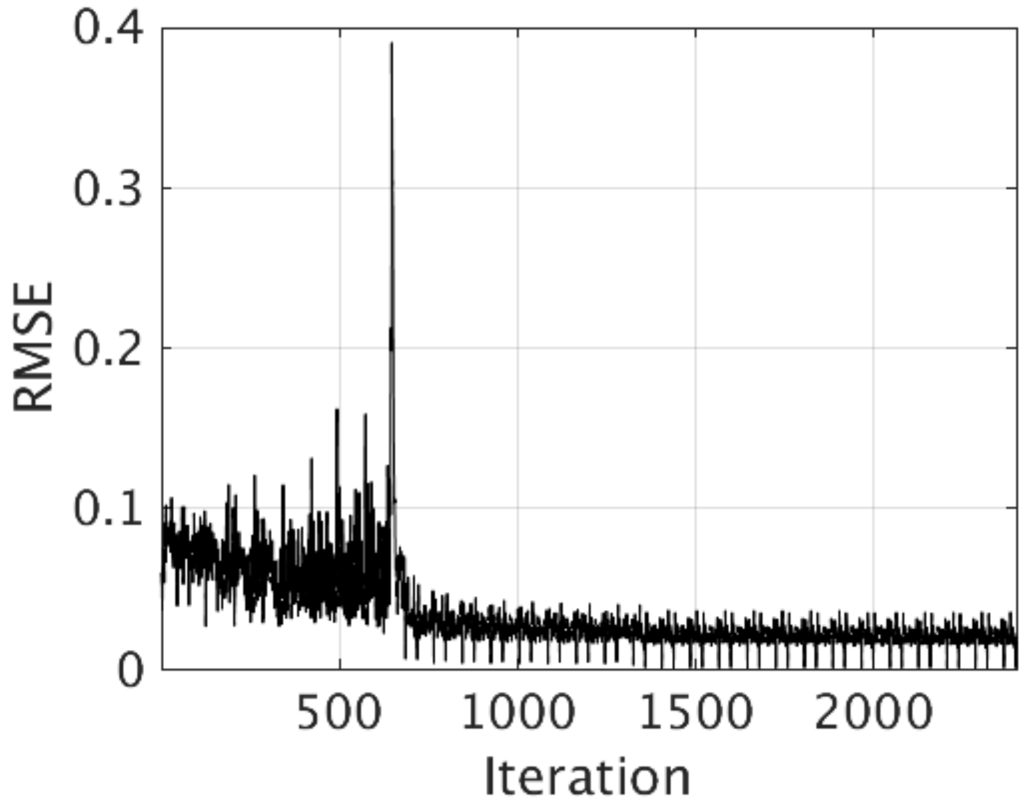

Output 42
0.2
0.1
0
−0.1
−0.2
0
0.05
0.1
0.15
0.2
Time(s)

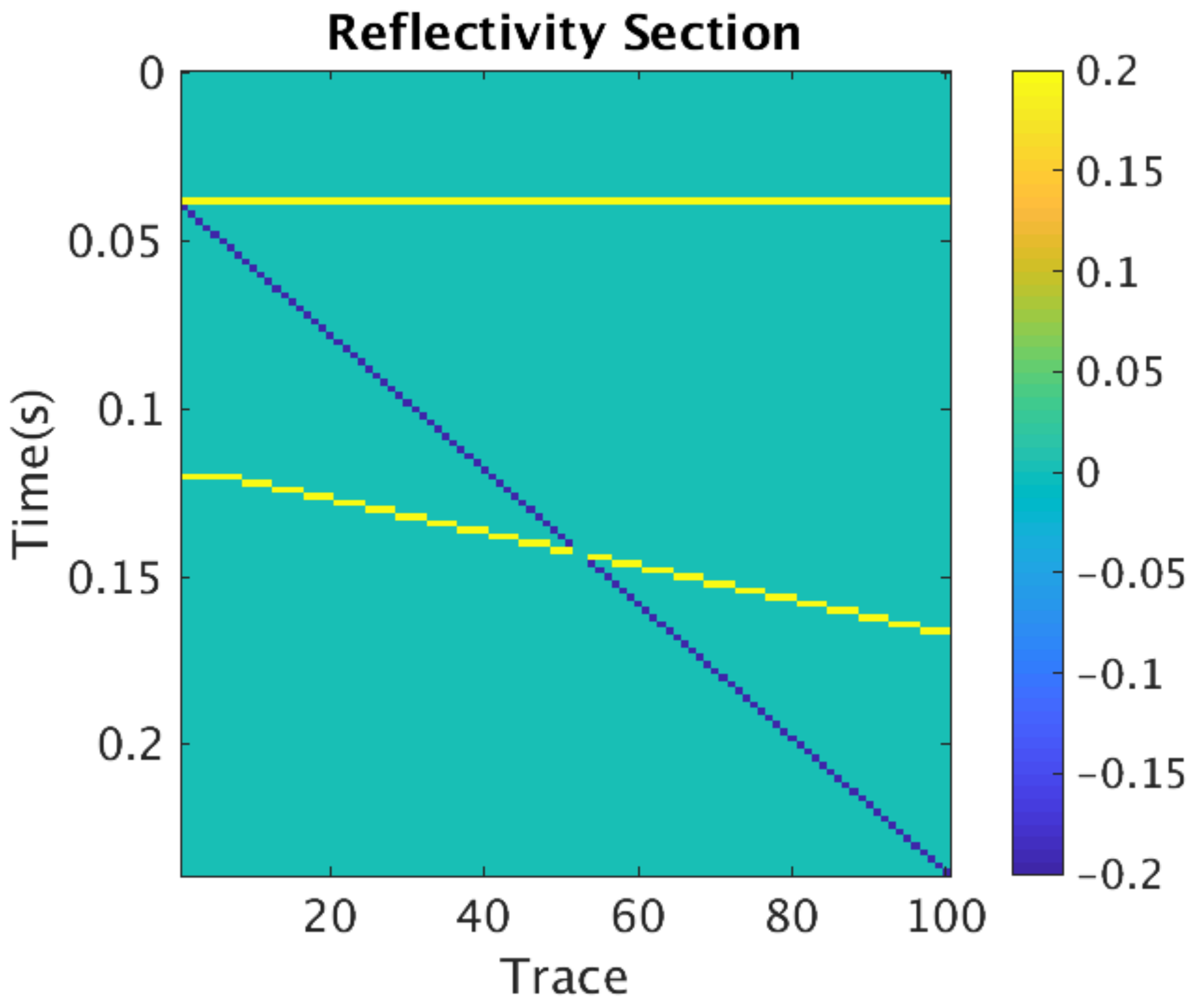

Input 42
0.4
0.2
0
−0.2
−0.4
0
0.05
0.1
0.15
0.2
Time(s)

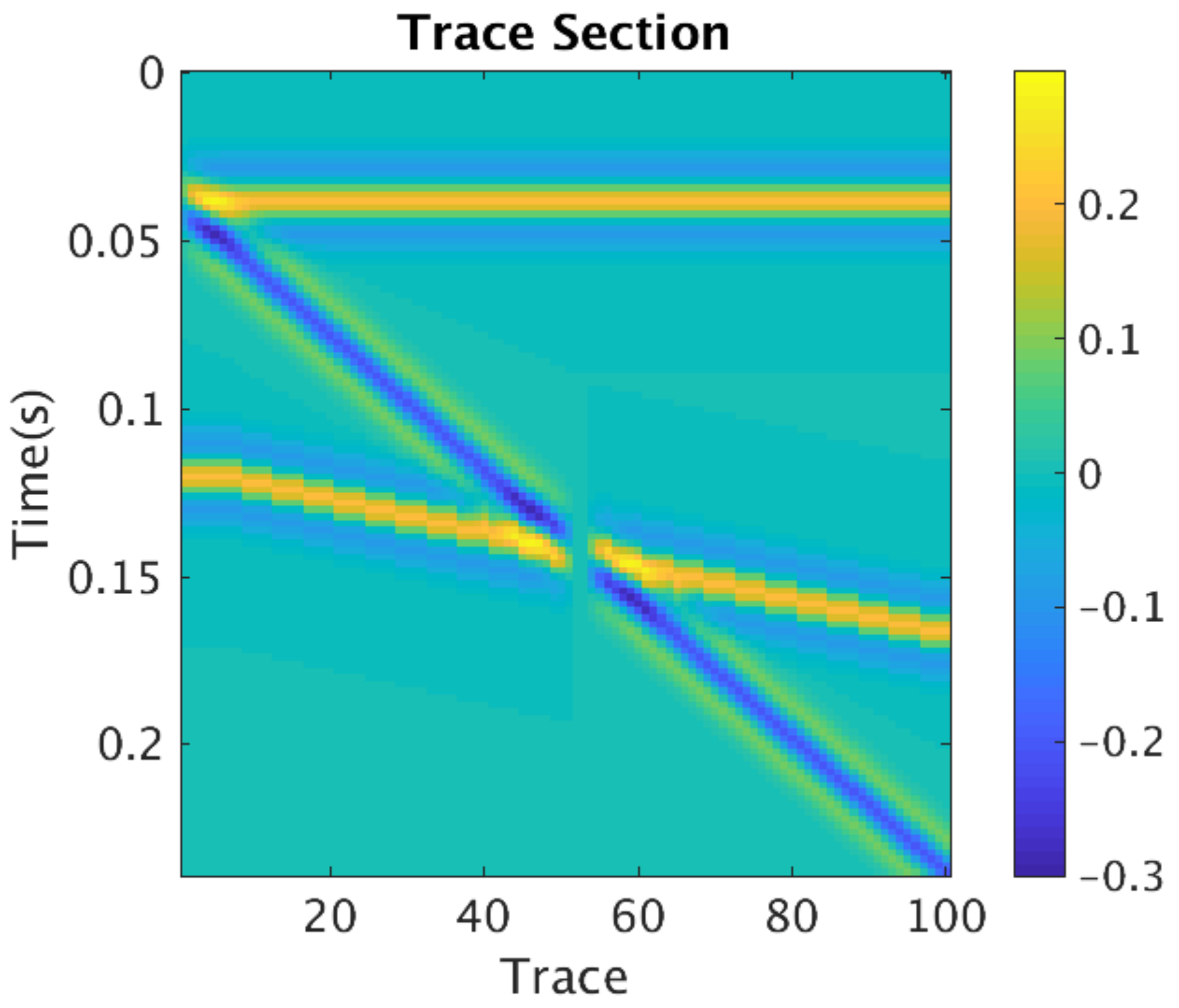
Convolution forward problem



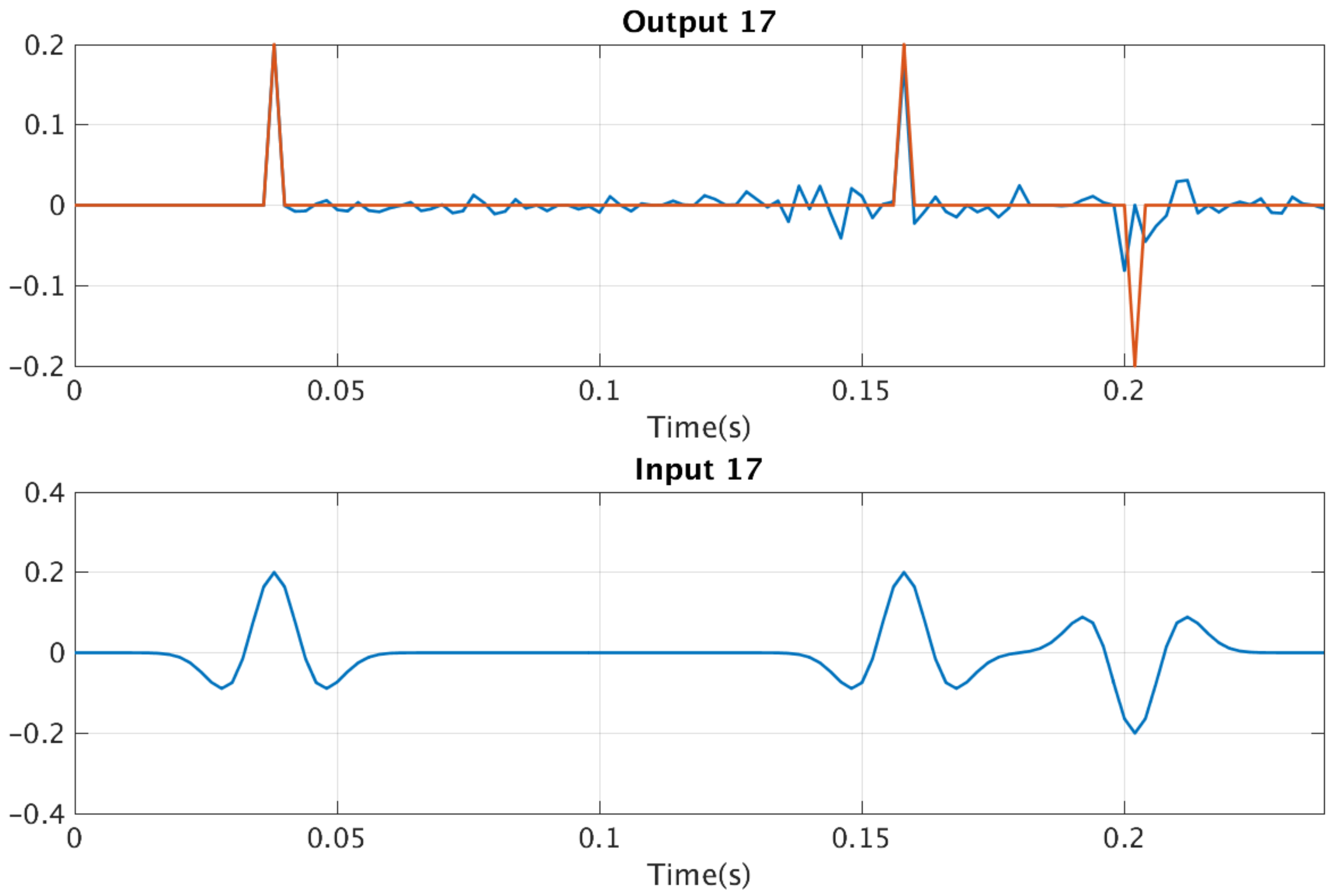
CNN



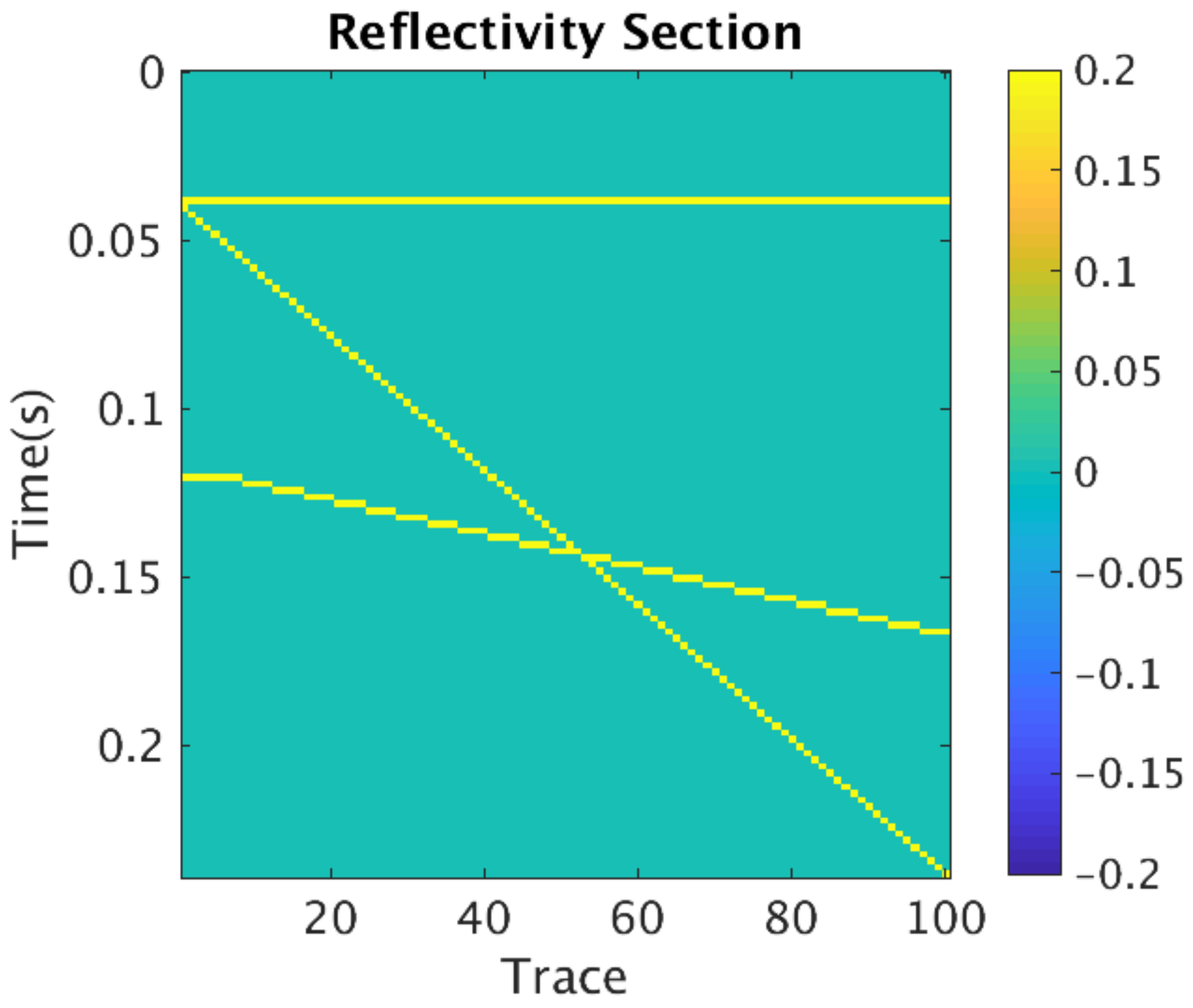

Reflectivity Section
0
0.05
0.1
0.15
0.2
Time(s)
20
40
60
80
100
Trace
0.2
0.15
0.1
0.05
0
−0.05
−0.1
−0.15
−0.2

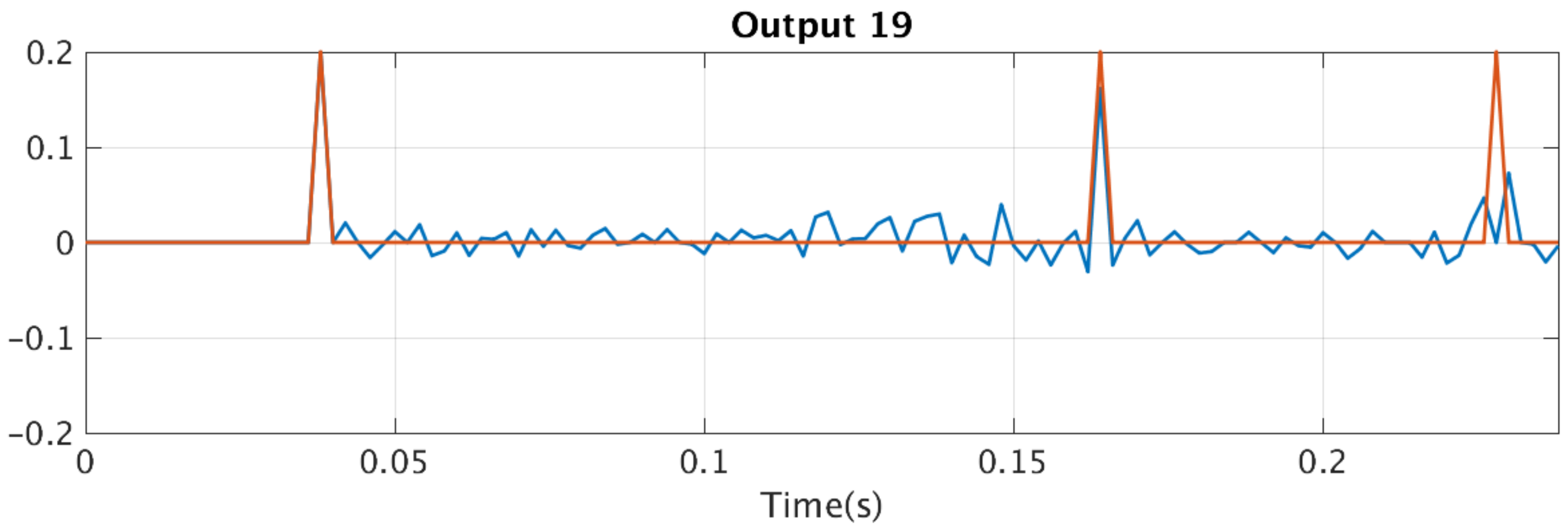
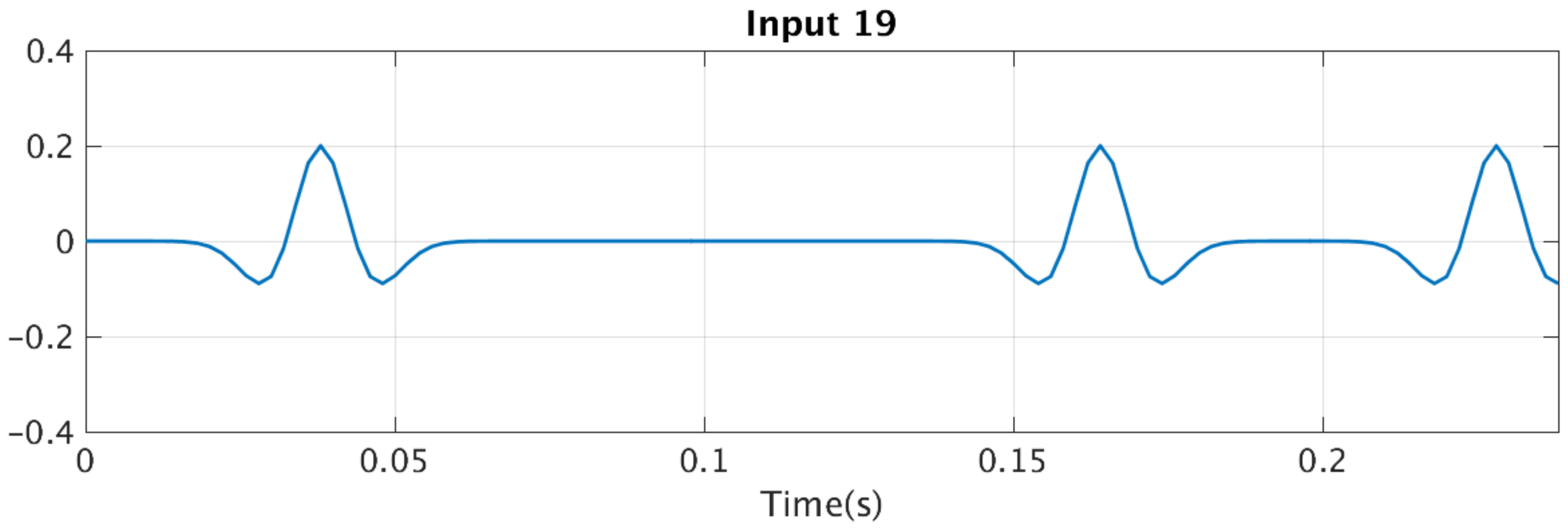

Trace Section
0
0.05
0.1
0.15
0.2
Time(s)
20
40
60
80
100
Trace
0.2
0.1
0
−0.1
−0.2
−0.3

Test

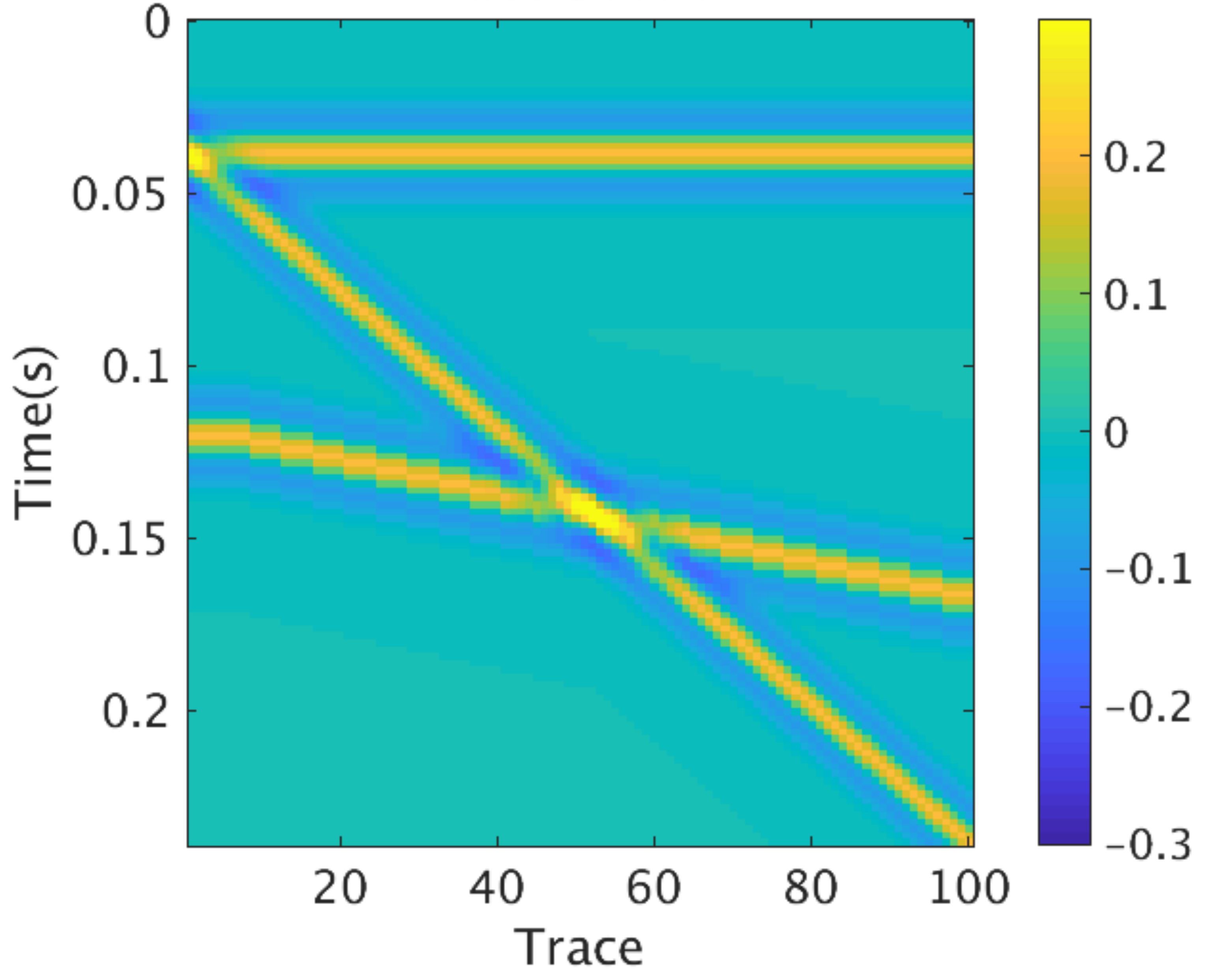


Output 17
0.2
0.1
0
−0.1
−0.2
0
0.05
0.1
0.15
0.2
Time(s)
Input 17
0.4
0.2
0
−0.2
−0.4
0
0.05
0.1
0.15
0.2
Time(s)


Reflectivity Section
Time(s)
Trace
0
0.05
0.1
0.15
0.2
20
40
60
80
100
0.2
0.15
0.1
0.05
0
−0.05
−0.1
−0.15
−0.2


Output 19
0.2
0.1
0
−0.1
−0.2
0
0.05
0.1
0.15
0.2
Time(s)

Input 19
0.4
0.2
0
−0.2
−0.4
0
0.05
0.1
0.15
0.2
Time(s)

Trace Section



Hopefield Neural Network (HNN) and Mean Field Annealing (MFA)

- A Hopfield network is a single-layer feedback network whose dynamics are governed by a system of nonlinear ordinary differential equations and by an energy function.
- A basic artificial neuron has many inputs and weighted connections.

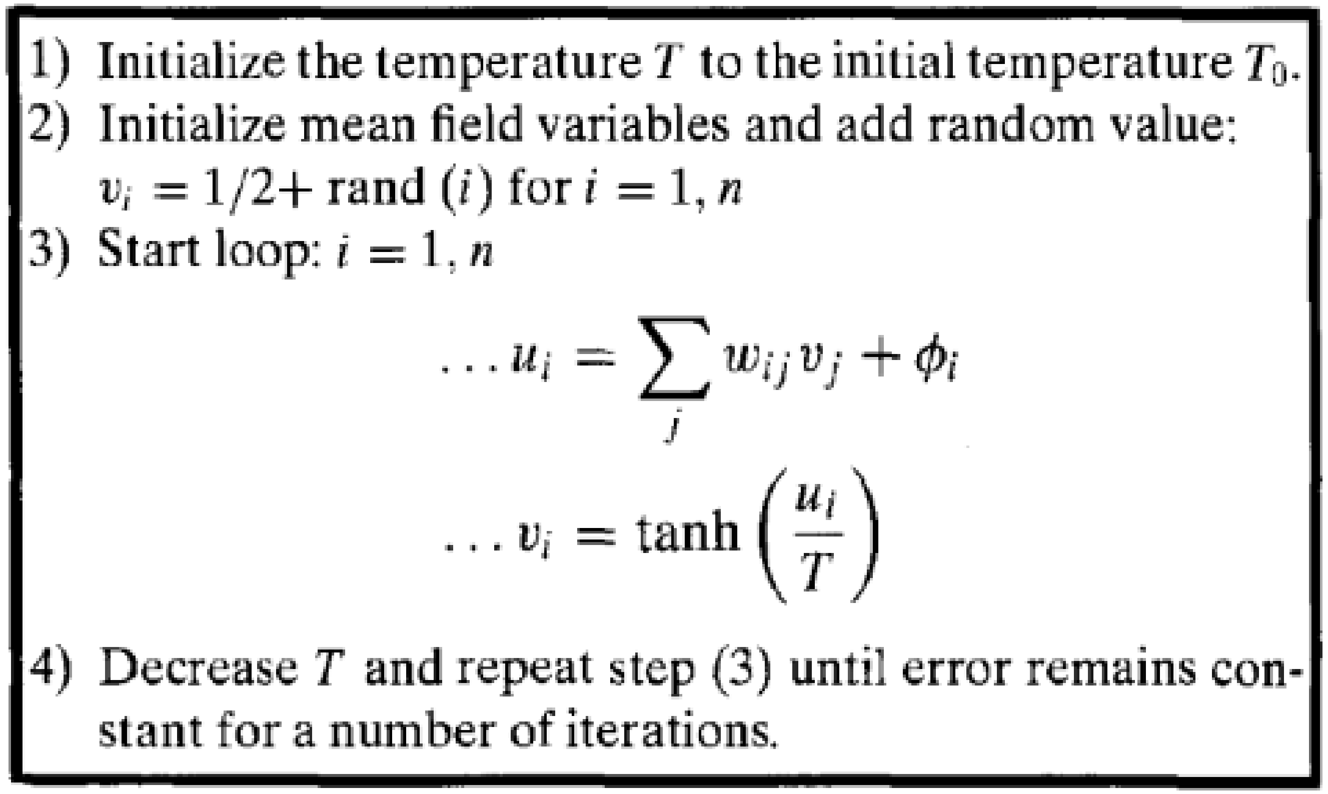
$$x_i(t+1) = H\left(\sum_{i \neq j=1}^{n} w_{ij} x_j(t) + \phi_i\right)$$

- where x_i can take the values 1 (on) or 0 (off) and H is the step function known as the activation function of the network.

Hopefield Neural Network (HNN) and Mean Field Annealing (MFA)

- Energy function $$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{i\neq j=1}^{n} w_{ij}x_i x_j - \sum_{i=1}^{n}\phi_i .$$
- The term energy function comes from a physical analogy to magnetic systems. In physics, the above equation is described as the Ising Hamiltonian of n interacting spins
- A central property of the Hopfield network is that given a starting point for the neurons, the energy will never increase as the states of the neurons change provided that w is a symmetric matrix with zero diagonal elements.
- Thus, one of the most important uses of a Hopfield network is in an optimization problem in which the cost function of the optimization problem is related to the energy function.

(Calderon-Macias and Sen)



```
1) Initialize the temperature $T$ to the initial temperature $T_0$.
2) Initialize mean field variables and add random value:
   $v_i = 1/2 + \text{rand}(i)$ for $i = 1, n$
3) Start loop: $i = 1, n$
       $\ldots u_i = \sum_j w_{ij} v_j + \phi_i$
       $\ldots v_i = \tanh\left(\frac{u_i}{T}\right)$
4) Decrease $T$ and repeat step (3) until error remains con-
   stant for a number of iterations.
```

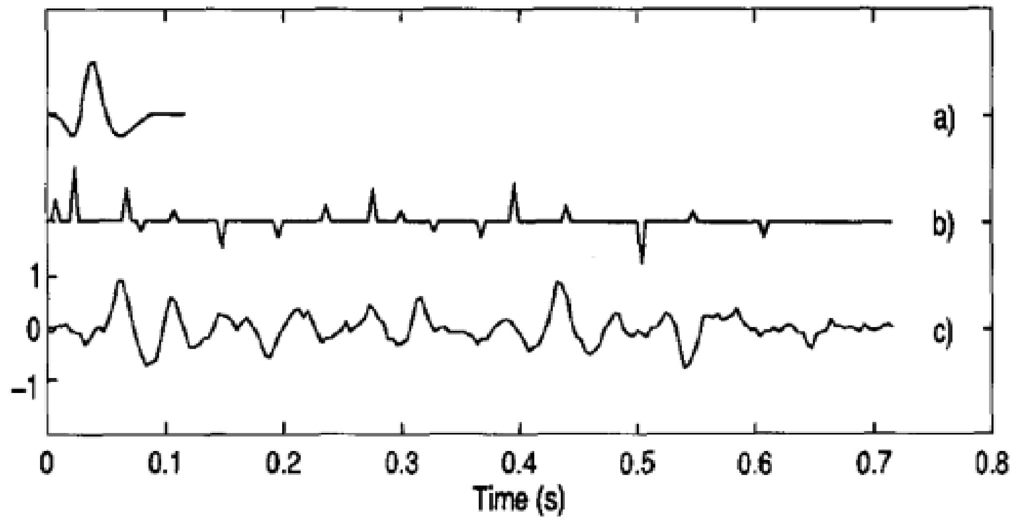
FIG. 1. Mean field annealing algorithm.

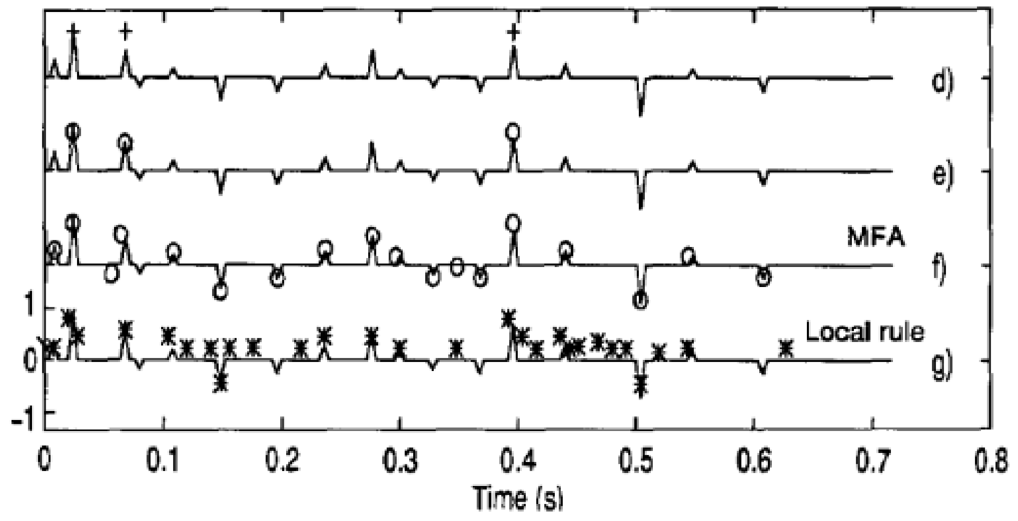
The reflectivity estimator is defined by two Hopfield networks:

- the first network detects time positions
- of a specific amplitude in the trace and

- the second refines the amplitude of these positions

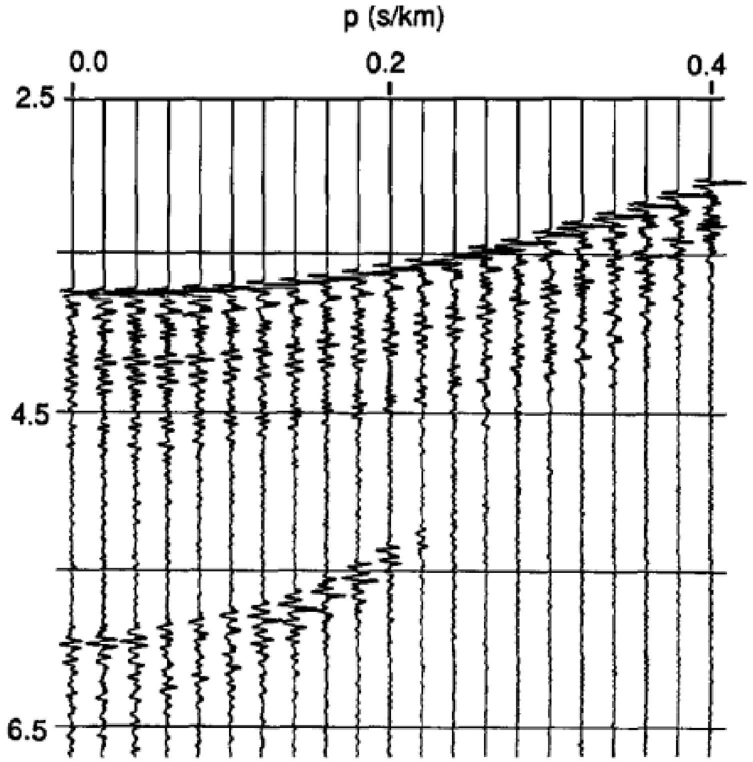
Synthetic Data Example

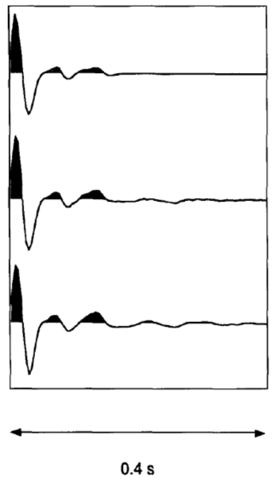


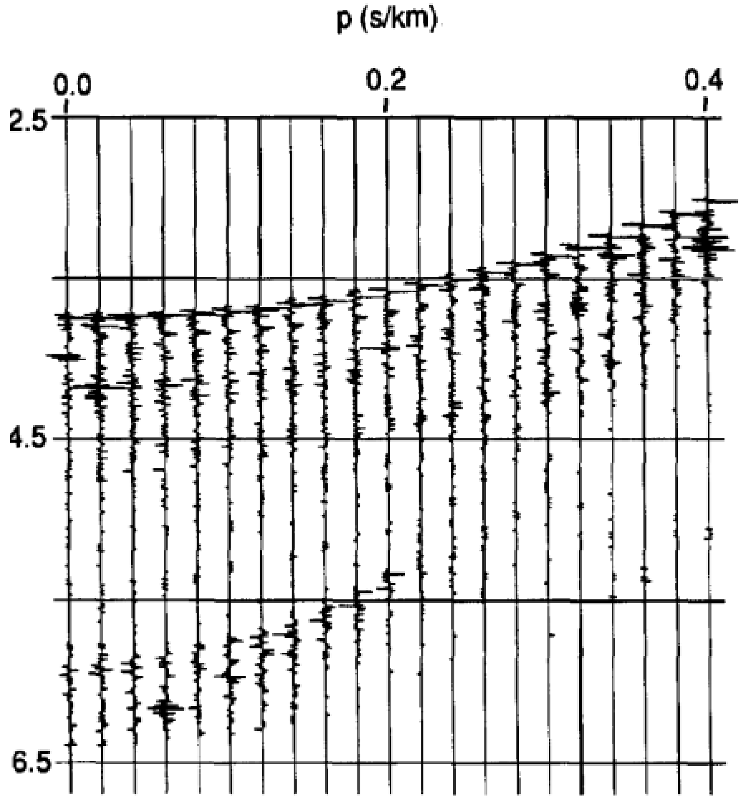


- Synthetic experiment: (a) Wavelet; (b) reflectivity series;
- (c) synthetic seismogram from convolving (a) and (b) with8% of noise added;
- (d) detected spike positions (+ symbol)for an amplitude of one using MFA. In solid line the true reflectivityis plotted and is the same in all subsequent plots;
- (e)computed amplitudes from (b) using MFA (o symbol);
- (f) computed reflectivity series obtained with MFA (o symbol) after sweeping amplitudes from [-1 1];
- (g) final reflectivity series obtained with the Hopfield local update rule (* symbol).

Real Data







Unsupervised learning

Facies analysis

Artificial immune-based self-organizing maps for seismic-facies analysis (Saraswat and Sen)

- Seismic facies analysis is the term sed for the process of extraction and interpretation of useful information related to seismic reflection parameters including geometry, envelope, continuity, and coherence. (Mitchum, 1977)
  - cluster and discriminant analysis,
  - Bayesian classification,
  - neural network, and
  - support vector machines.

Saggaf et al. (2003) , Li and Castagna (2004), Jin, Sen and Stoffa (2009), Taner et al., 2001; Matos et al., 2003a, 2003b, 2004, Roy et al., 2010).

Self Organizing Maps (SOM)

- Application of SOM to seismic facies analysis requires utmost care because noise in the data set may give rise to unrealistic features.
- Thus, an SOM-based facies map derived from noisy seismic data may not be representative of subsurface geologic features (Matos et al., 2007).
- SOM is very memory intensive (Kurdthongmee, 2008).
- We first subject the seismic data to an artificial immune system (AIS), which is a robust data-clustering procedure that also reduces dimensionality.

We use a hybrid trace waveform classification algorithm which combines AIS and SOM, termed AI-SOM, that takes advantage of these algorithms. In particular, the numbers of classes or clusters in a data set are initially detected using an artificial immune system, which results in reduction in dimensionality. This is followed by classification of facies using self-organizing maps (Roy et al., 2010), facilitating extraction of high-level information from clusters in the data.

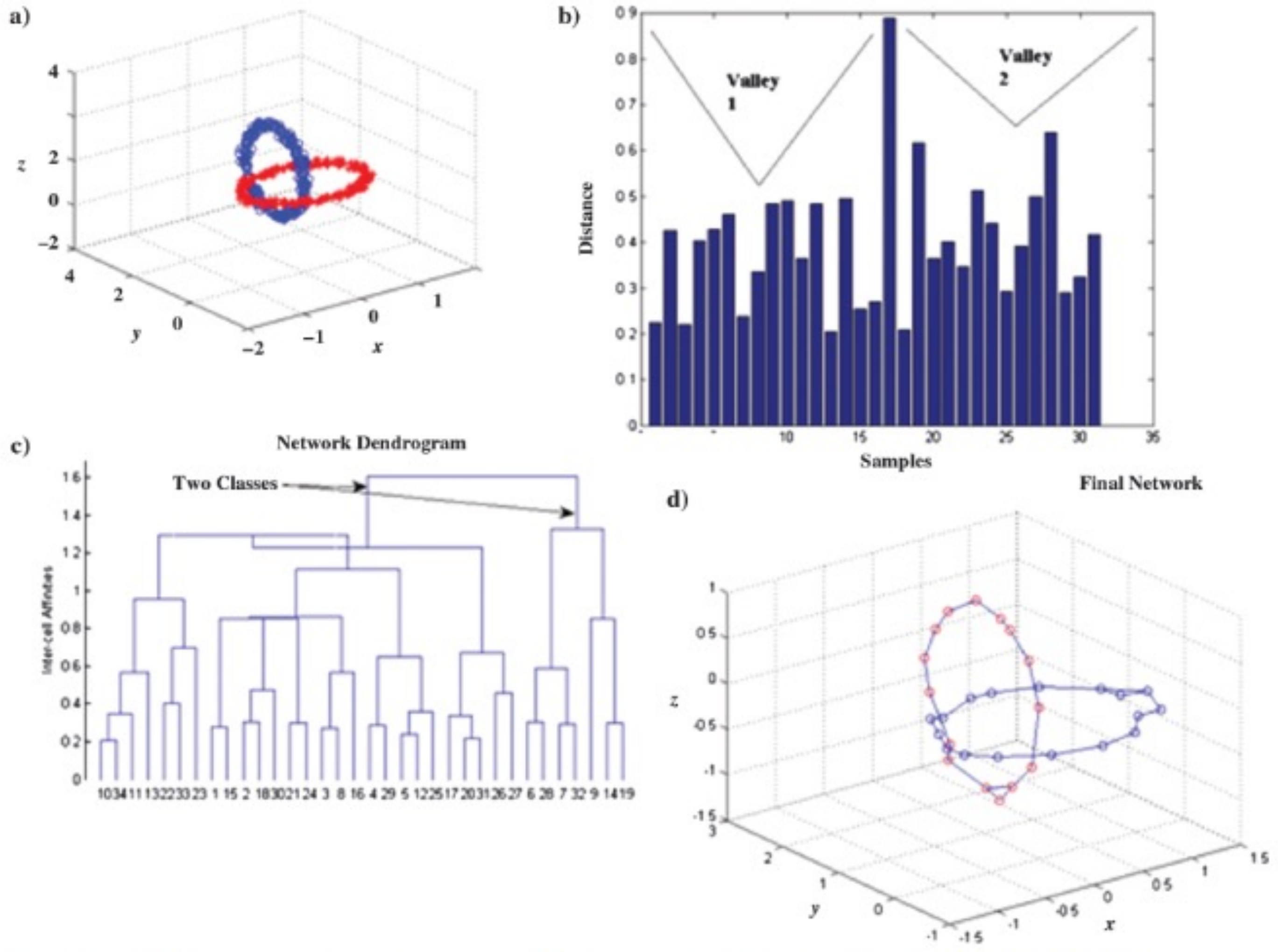


Figure 2. Artificial immune network classification: (a) Input data set with two classes of chain links (from de Castro 2002); (b) histogram plot of the memory matrix where valleys are identified as number of clusters; (c) network dendrogram showing two major classes with further subdivision and their intercell affinities; (d) final network structure, i.e., reduced data set.

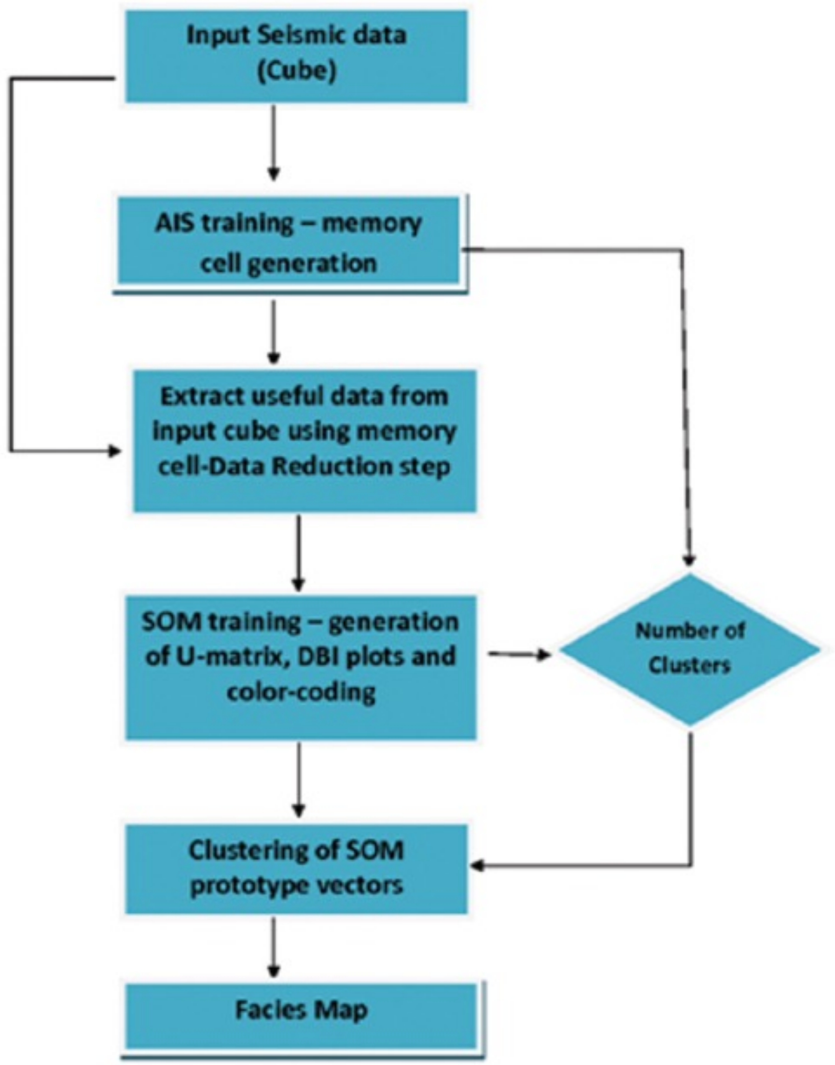

Input Seismic data (Cube)
AIS training – memory cell generation
Extract useful data from input cube using memory cell-Data Reduction step
SOM training – generation of U-matrix, DBI plots and color-coding
Number of Clusters
Clustering of SOM prototype vectors
Facies Map

Figure 5. Flowchart representing workflow of AI-SOM method for seismic facies analysis.

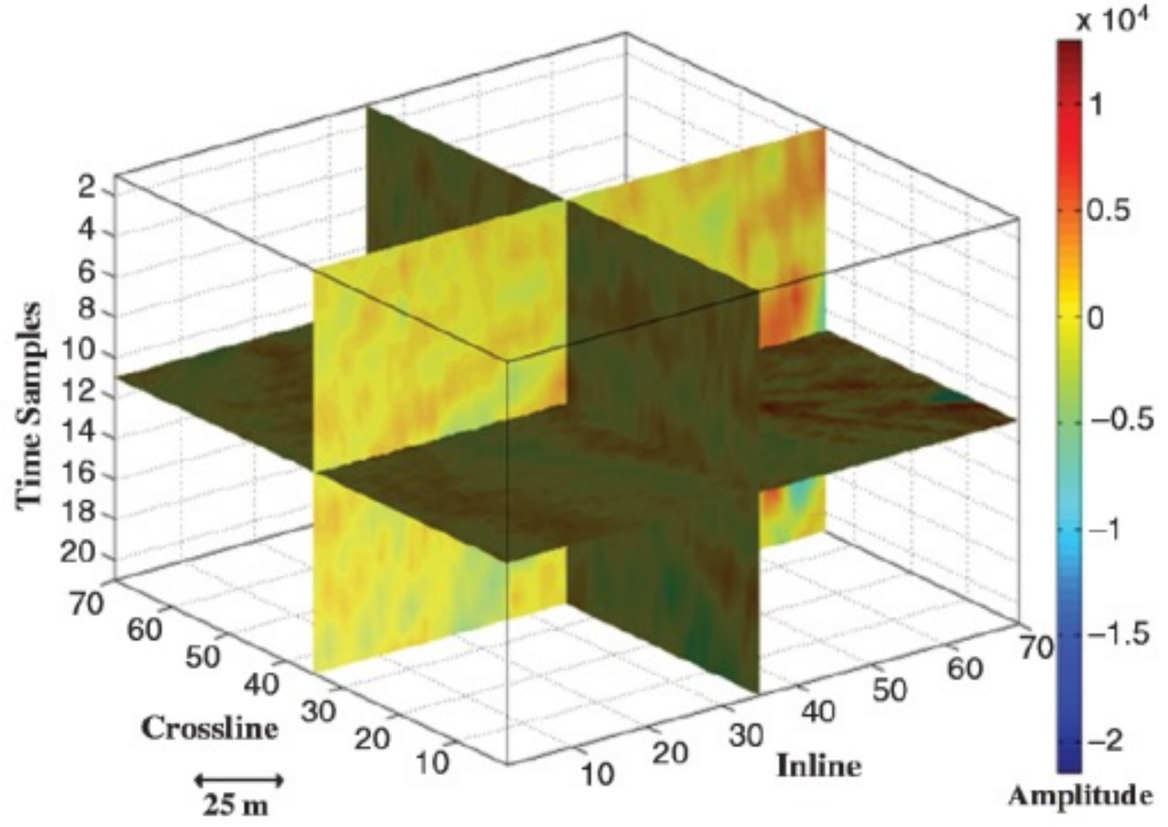


Figure 7. Input seismic data to AIS algorithm to prove the efficacy of AIS for data reduction.

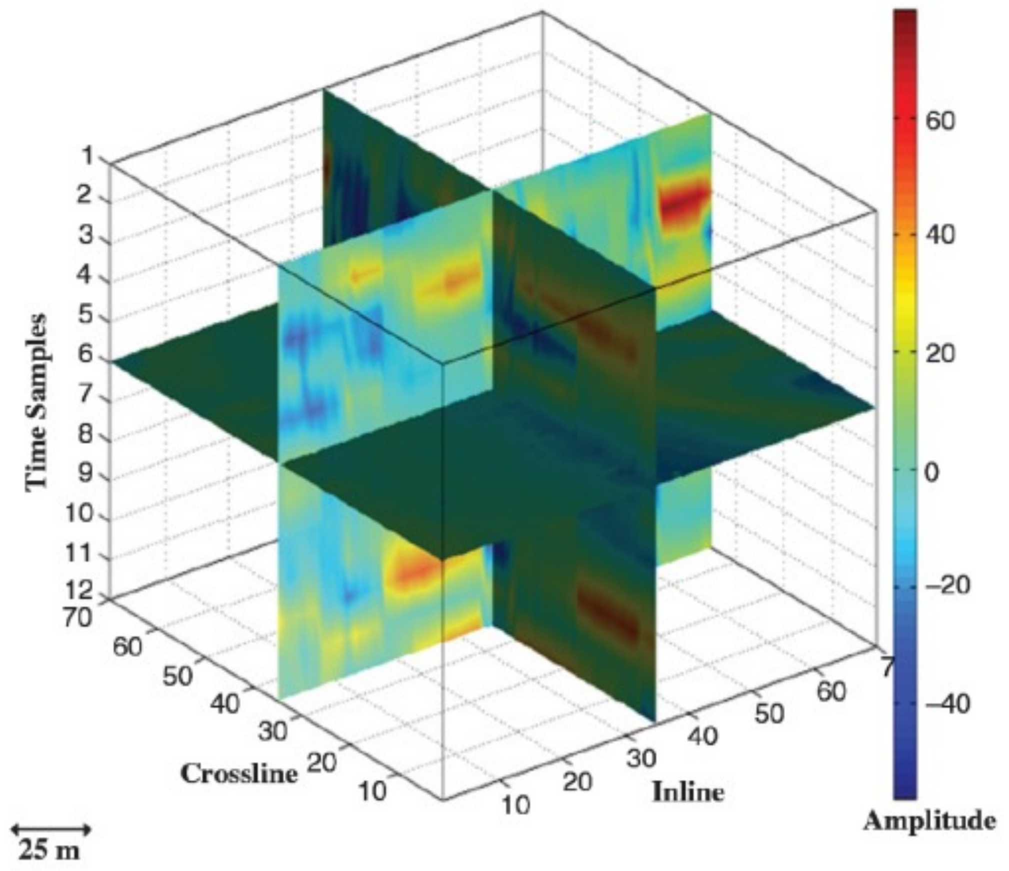


Figure 8. Result from AIS over input seismic data shown in Figure 6. The dimensionality is reduced from $(70 \times 70 \times 20)$ to $(70 \times 70 \times 12)$. It is also evident from the result that most of the features are preserved while reducing the data set.

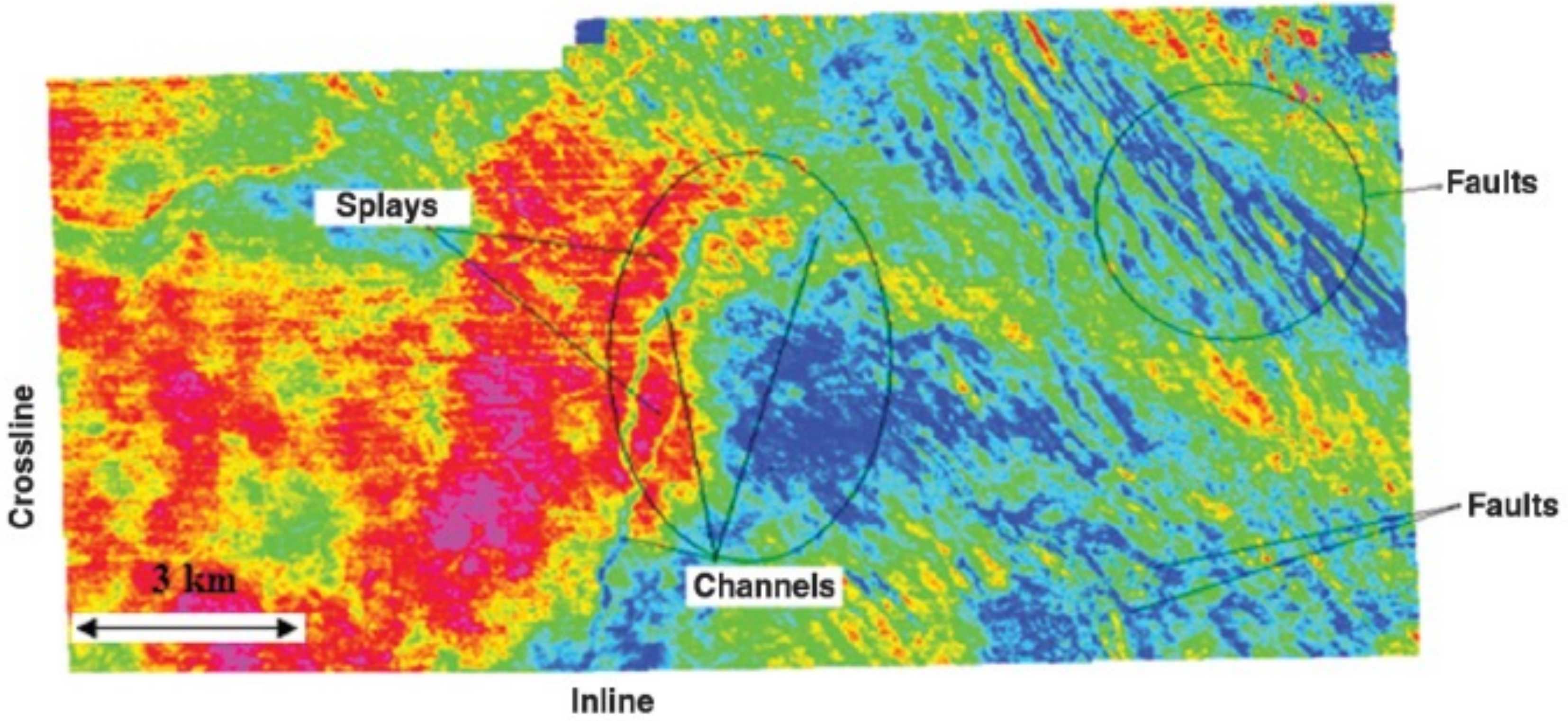


Figure 11. Final facies map generated after SOM clustering; various depositional and structural features such as channels, splays, and faults are identified. The color code for different clusters is shown in Figure 10c.

Using “Feature Selection” to Improve the Probabilistic *K-nn* Bayesian Classification Method

Barone and Sen, SEG 2017

Background

- A data classification method that is designed to estimate pore fluid type.
- In practice, this method will use Vp, Vs, density, porosity and pore fluid type information from well logs to estimate pore fluid type in an earth model where Vp, Vs and density have already been estimated by seismic inversion.
- Our method is basedon the probabilistic nearest neighbor (k-nn) method, but also incorporates feature selection and nonlinear regression.
- To improve upon the standard k-nn method, we incorporate a regression approach based on the Bayesian MARS model to estimate porosity in the earth model.
- Additionally, we utilize a feature selection method that automatically generates various data combinations and evaluates their usefulness for classification.

Background

- A Bayesian approach to multivariate adaptive regression spline (MARS) fitting (Friedman, 1991) is used.
- This takes the form of a probability distribution over the space of possible MARS models which is explored using reversible jump Markov chain Monte Carlo methods (Green, 1995; Sen and Biswas 2017).

Background

Definitions:

1. Training Data: a subset of the data where data class (e.g., pore fluid type) is known in addition to one or more indicator variables (e.g., V_p, V_s, and ρ)
2. Test Data: the remainder of the data where the indicators are known but the data class is not. The goal is to "solve" for the class using the indicator data.

The *Probabilistic k-nn* algorithm:

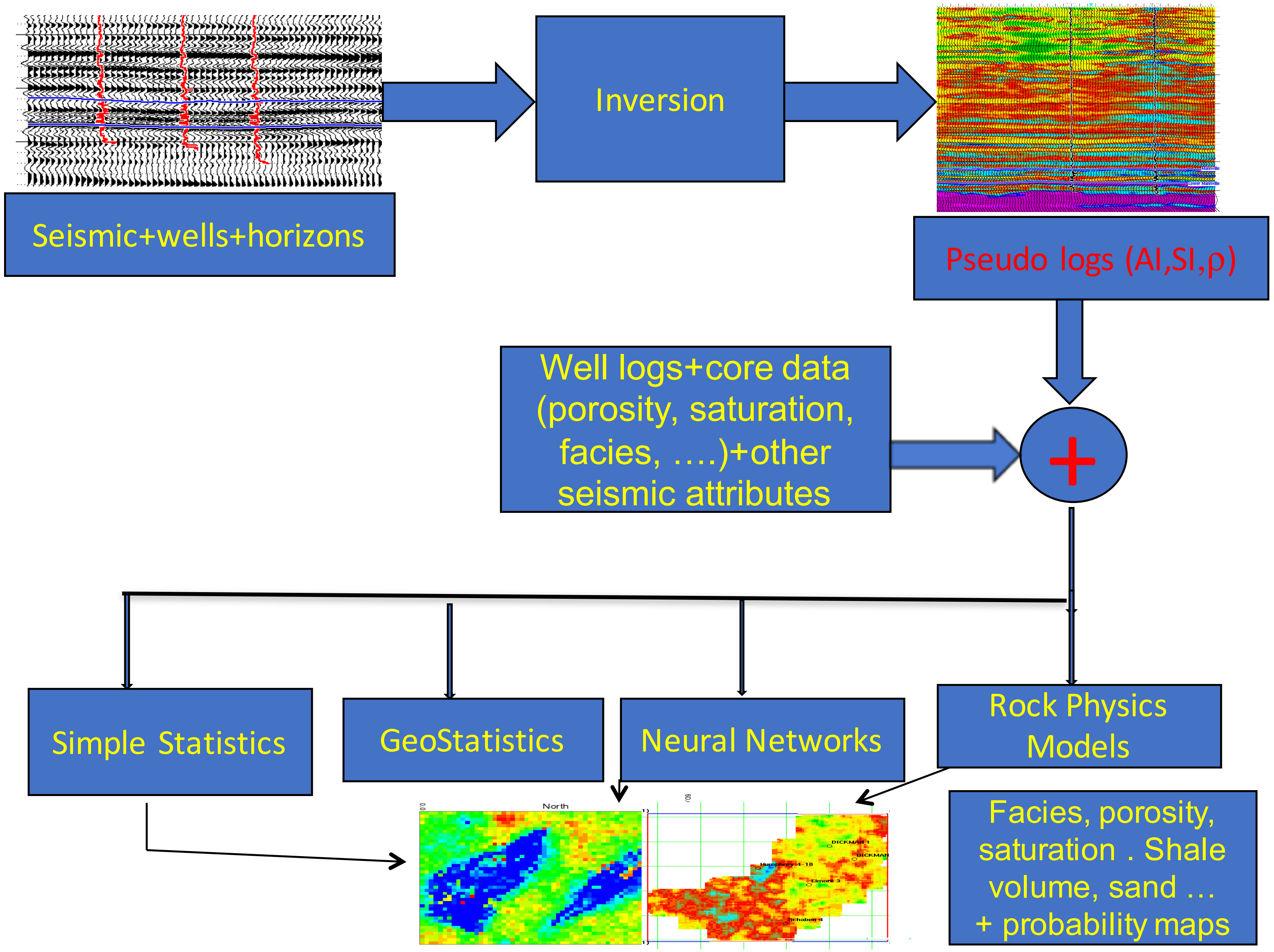
1. Calculate a "distance" between each combination of indicators in the Training Data (where class is known) and each combination of indicators in the Test Data (where class is not known).
2. For each data point in the Test Data, choose *k* points from the Training Data with the smallest "distance" .
3. Perform a Reverse-Jump Markov-Chain Monte-Carlo (RJMCMC) analysis using the *k* nearest points and uninformative priors. By recording how often each of these points (and the data classes they represent) are accepted during the RJMCMC run, one can determine the probability of each data class at every point in the Test Dataset.

New Method

GOAL: take the "Probabilistic *k-nn*" method (Holmes and Adams, 2002; Denison et al., 2002.) and **improve it by incorporating elements of "Feature Selection"**:

1. Automatically combining and perturbing inputs using addition, subtraction, multiplication, division, squaring, and square root operations.
2. Automatically calculate a measure of "how well the classes are separated" in each domain using the training data (*overall separation* and *locally separated by class and porosity*)
3. Select the best *overall separated* domains to use as input into the probabilistic k-nn algorithm
4. Modify the RJMCMC part of the *k-nn* algorithm so that inputs are weighted based on *local separation (a function of class and porosity)*

SEE ANTONY BARONE'S TALK AT SEG


Seismic+wells+horizons
Inversion
Pseudo logs (AI,SI,ρ)
Well logs+core data (porosity, saturation, facies, ….)+other seismic attributes
+
Simple Statistics
GeoStatistics
Neural Networks
Rock Physics Models
Facies, porosity, saturation . Shale volume, sand … + probability maps
North
DICKMAN 1
DICKMAN
Elmore 3
Schaben 4

Summary

- We have several years of experience in working with machine learning applications to Geophysical problems
- Time to employ ML to seismic reservoir characterization
- Fractured reservoirs
- We need to work on hyperparameter selections – perhaps rjHMC
- Global optimization for training.