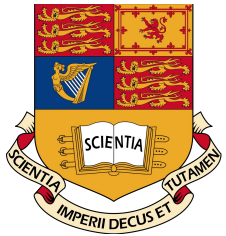


Imperial College London

Department of Earth Science and Engineering
MSc Applied Computational Science and Engineering

Module ACSE-9, IRP Presentation

Neural Networks Applied to Signal Separation on Multi-Sensor Array Data

Author:
Mattia Guerri

Supervisors:
Prof. Olivier Dubrule, Dr. Lukas Mosser

Company Supervisors (CGG):
Dr. Song Hou, Dr. Henning Hoeber

Presentation Outline

Mattia Guerri
MSc ACSE, 2018-2019

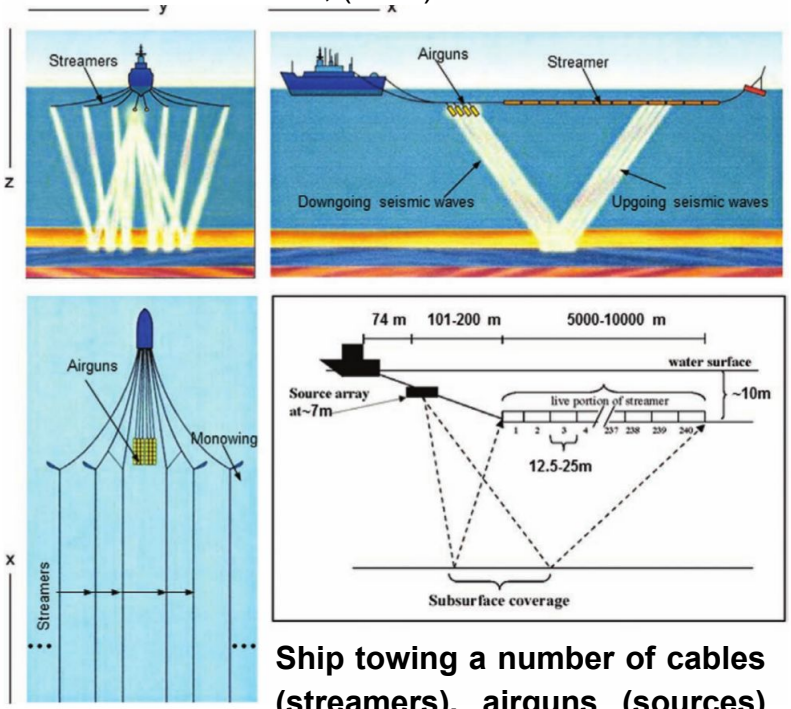
- Introduction:
  - Data acquisition and principal features.
  - Project goal.
  - Neural networks applied to image semantic. segmentation.
- Methodology:
  - Neural networks implementation.
- Results:
  - Impact of dataset variations.
  - Effects of networks architectural modification.
  - Limitations of the networks output.
- Conclusion:
  - List of the project major findings.

The project was conducted in form of an internship with CGG (Compagnie Générale de Géophysique). The company made available dataset and facilities.

Data Acquisition
Marine Towed Streamer Seismic Survey

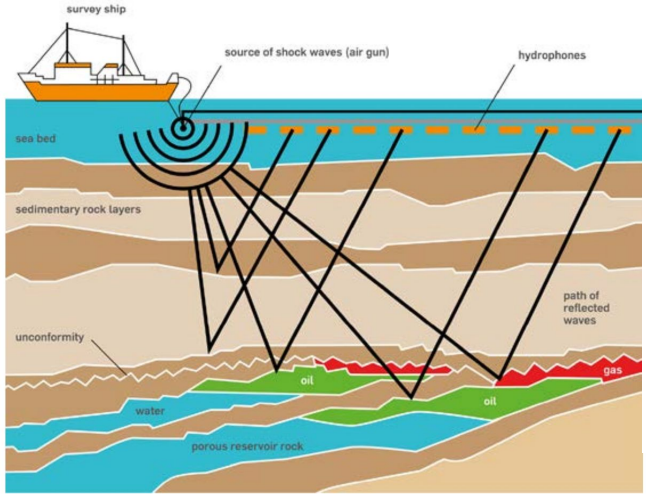
Mattia Guerri
MSc ACSE, 2018-2019

Ikelle and Amundsen, (2017)

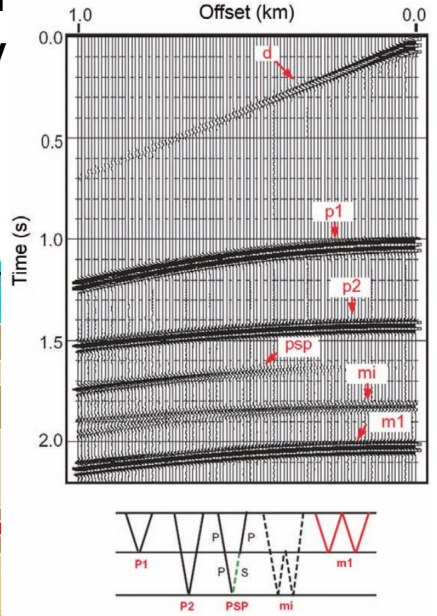


Ship towing a number of cables (streamers), airguns (sources) and hydrophones (receivers).

A method of primary importance in the investigation of sedimentary basin.



Waves generated by the source are reflectected and then recorded by the receivers.



Receivers recording are gathered to produce the dataset.

Data Acquisition
Mooltishooting Technique

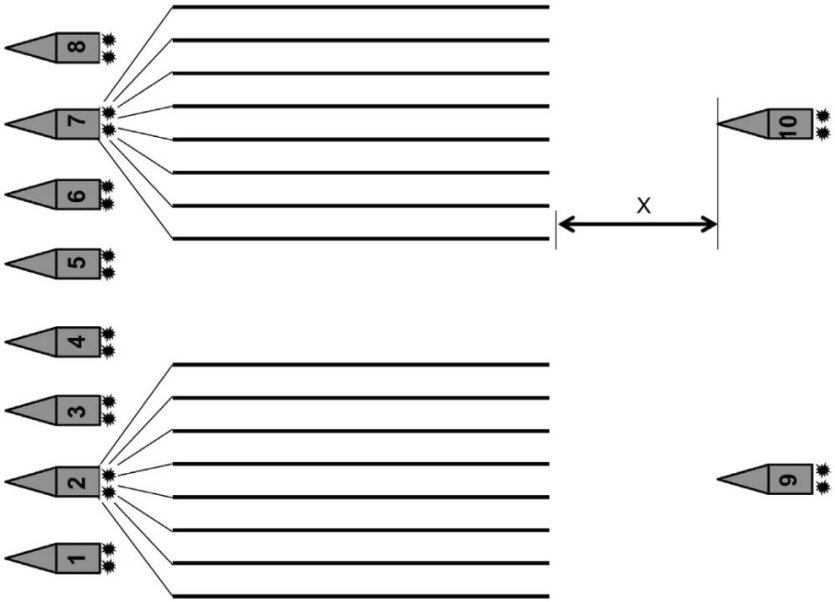
Mattia Guerri
MSc ACSE, 2018-2019

Multiple sources are fired simultaneously in the same acquisition area.

The goals of multishooting are:

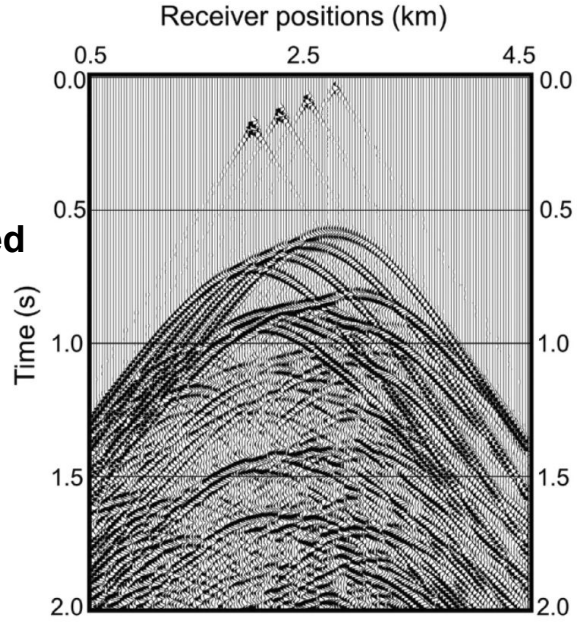
- **Reduction of the acquisition costs**
- **Obtaining a dataset of improved quality (azimuthal coverage)**

The acquired dataset is characterised by the interference of signal coming different sources.



Ikelle and Amundsen, (2017)

The signal in the blended dataset needs to be separated before processing.



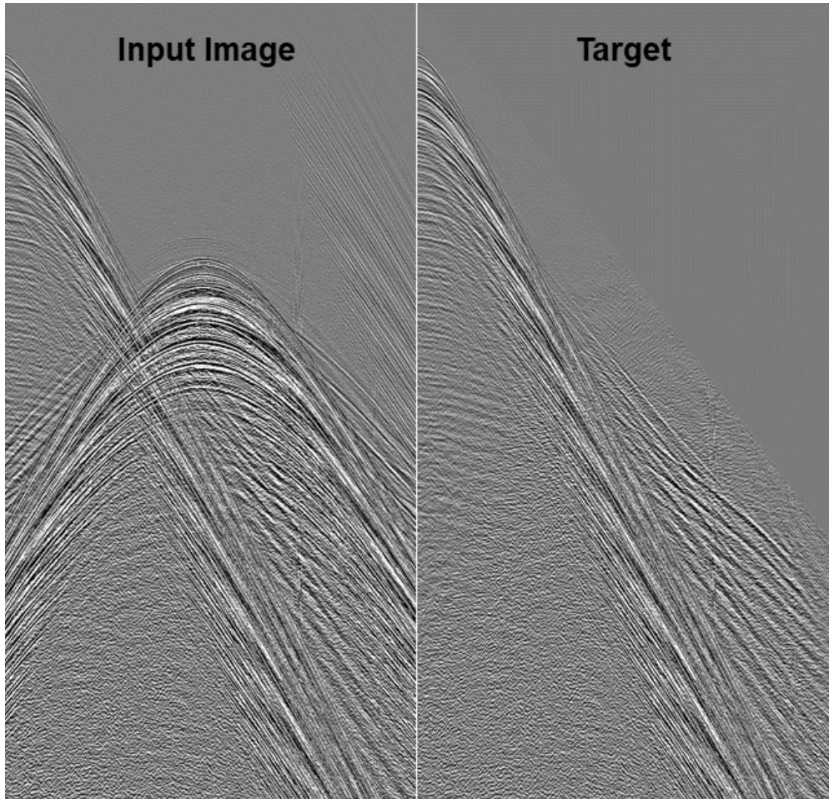
Project Goal
Apply Neural Networks to Signal Separation

Mattia Guerri
MSc ACSE, 2018-2019

Example of blended Dataset

The signal generated by one source develops from the edge of the image.

The second source generates the dome-like structure in the center of the image.



Example of de-blended signal (G2 Output)

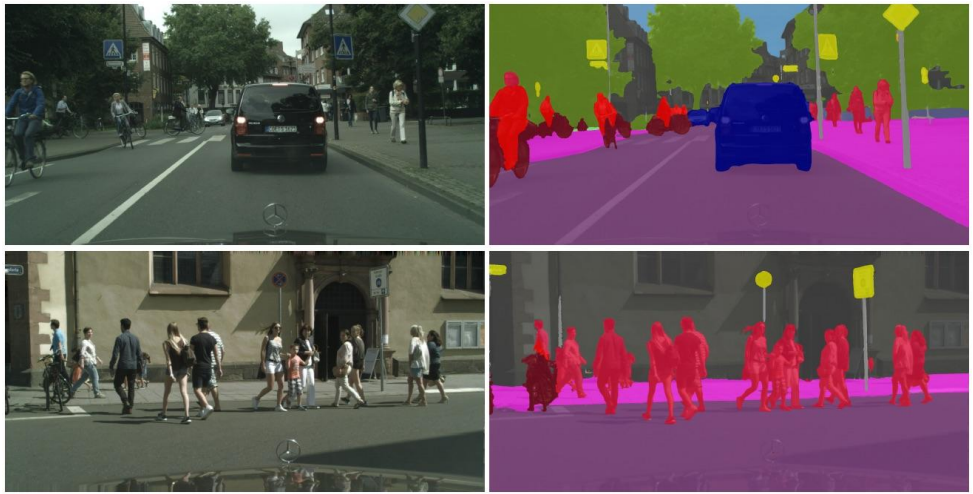
The signal generated by the second source has been removed.

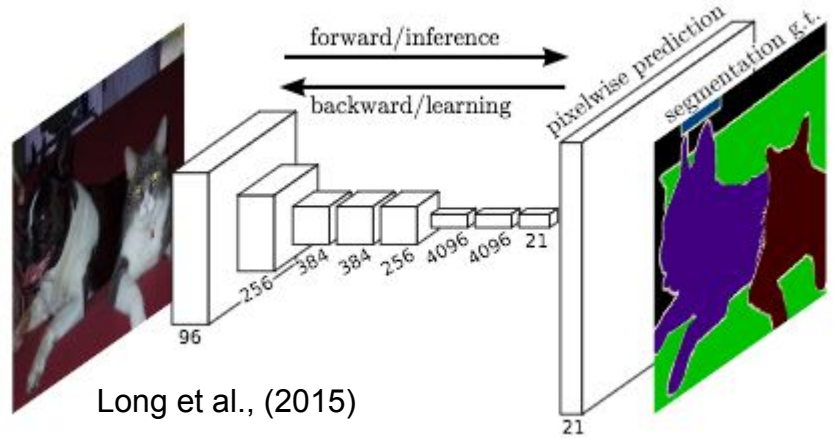
Signal de-blending is very expensive both in terms of time and computational resources.

The goal of the project is to develop and test neural networks capable of producing the de-blended signal.

Neural Networks in Image Processing
Image Semantic Segmentation

http://vladlen.info/publications/feature-space-optimization-for-semantic-video-segmentation/





Long et al., (2015)

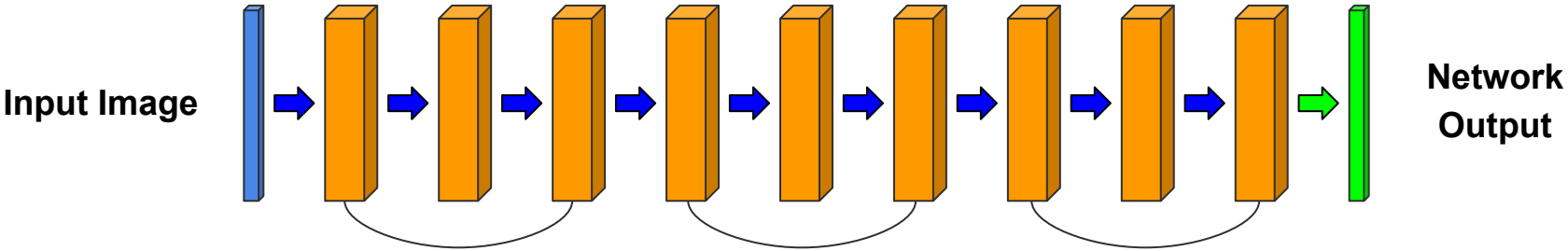
Image semantic segmentation shares similarities with the problem at hand.

Every pixel of the image is assigned to a certain class.

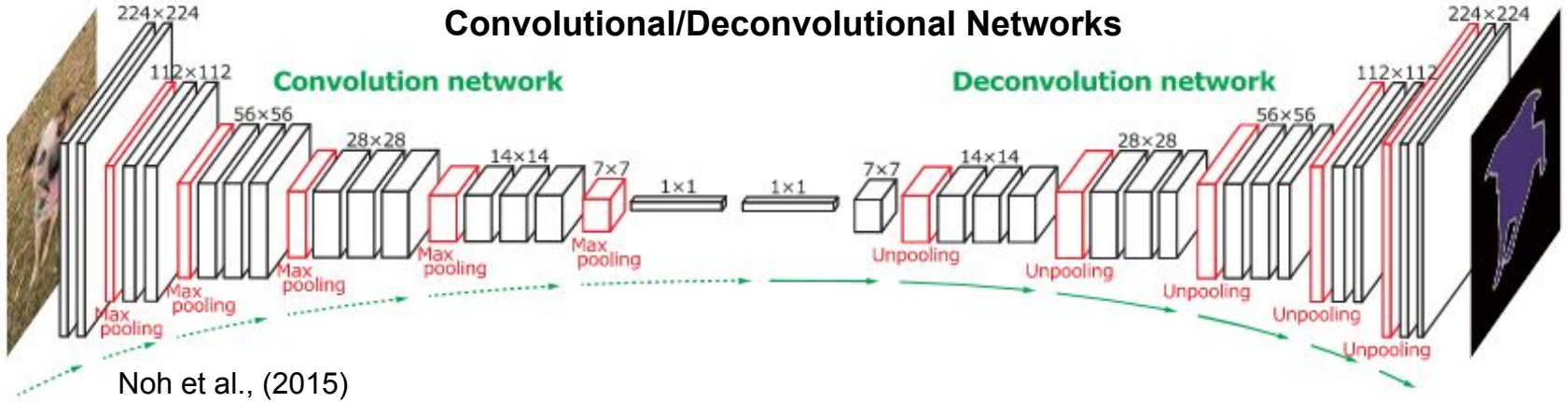
Mattia Guerri
MSc ACSE, 2018-2019

Image Semantic Segmentation
Neural Networks Architectures

Fully Convolutional Networks



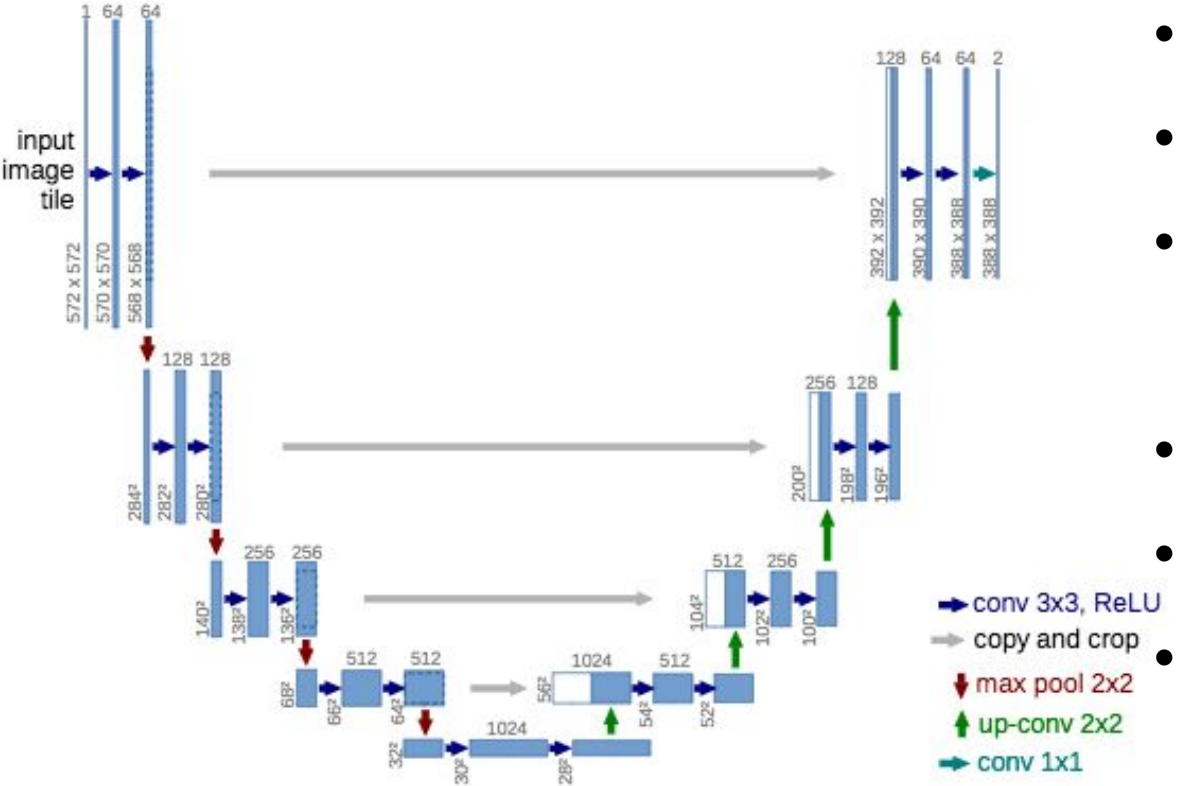
Convolutional/Deconvolutional Networks



Noh et al., (2015)

Mattia Guerri
MSc ACSE, 2018-2019

Image Semantic Segmentation
U-shaped Architectures

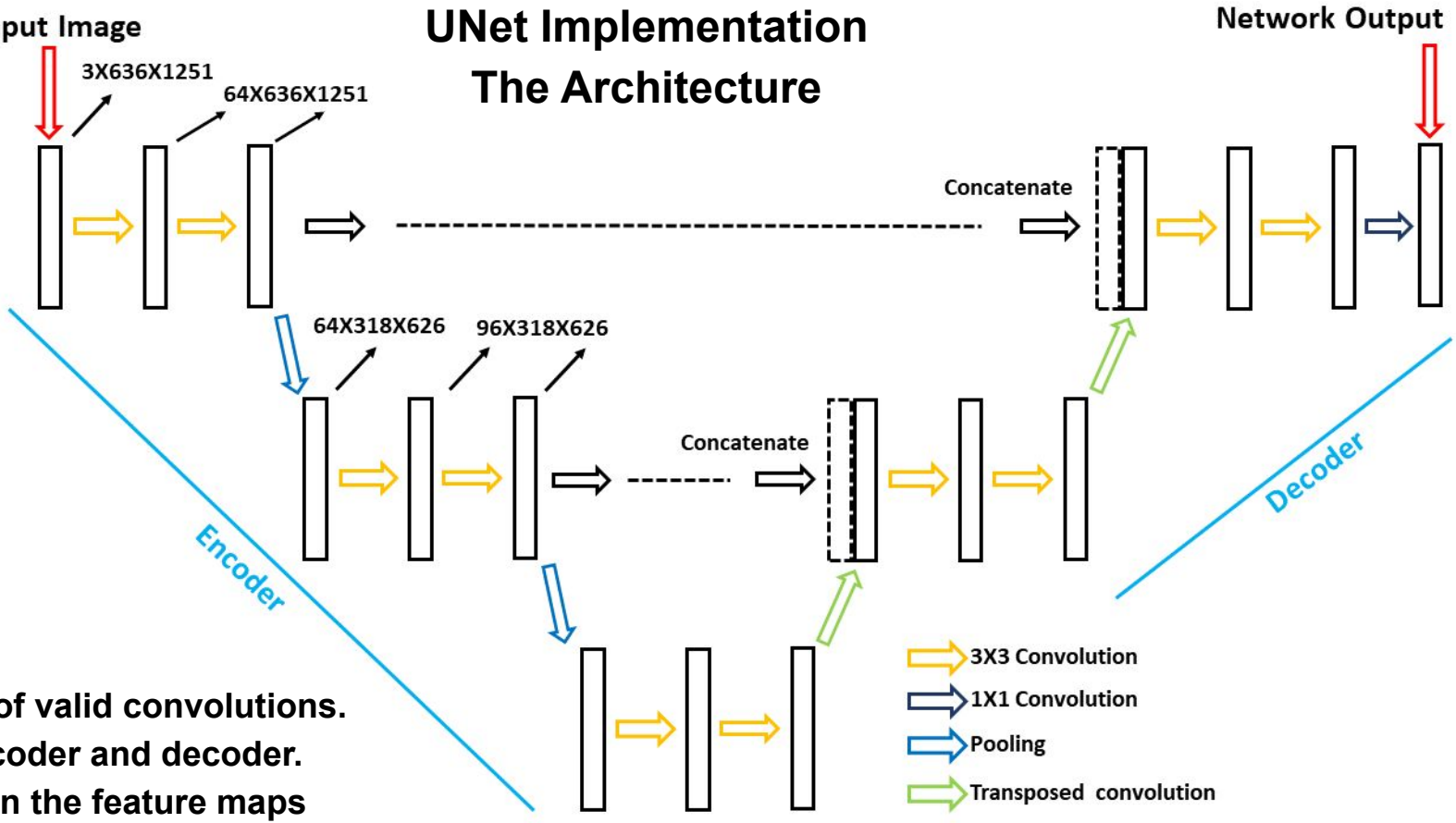


Ronneberger et al., (2015)

- Developed for processing high-resolution biomedical images.
- Characterised by an encoding and a decoding path.
- Outputs of the encoder are concatenated to the corresponding decoder outputs in order to improve localization.
- Each blocks consists in 2 convolutions (valid).
- Number of kernels per convolution doubles in each block.
- Input size must allow pooling of features maps with even size.

Mattia Guerri
MSc ACSE, 2018-2019

UNet Implementation
The Architecture



- Same instead of valid convolutions.
- Symmetric encoder and decoder.
- No limitation on the feature maps size (padding before pooling).
- Number of kernels and number of blocks is determined by the user.

Mattia Guerri
MSc ACSE, 2018-2019

UNet Implementation
The Code

Mattia Guerri
MSc ACSE, 2018-2019

- Code written in Python and using classes imported from the Pytorch framework.
- Network implementation structured in three classes.
- DownBlock: implements one block of the encoder.
- UpBlock: implements one block of the decoder.
- UNet: builds up the network instantiating the previous two classes.

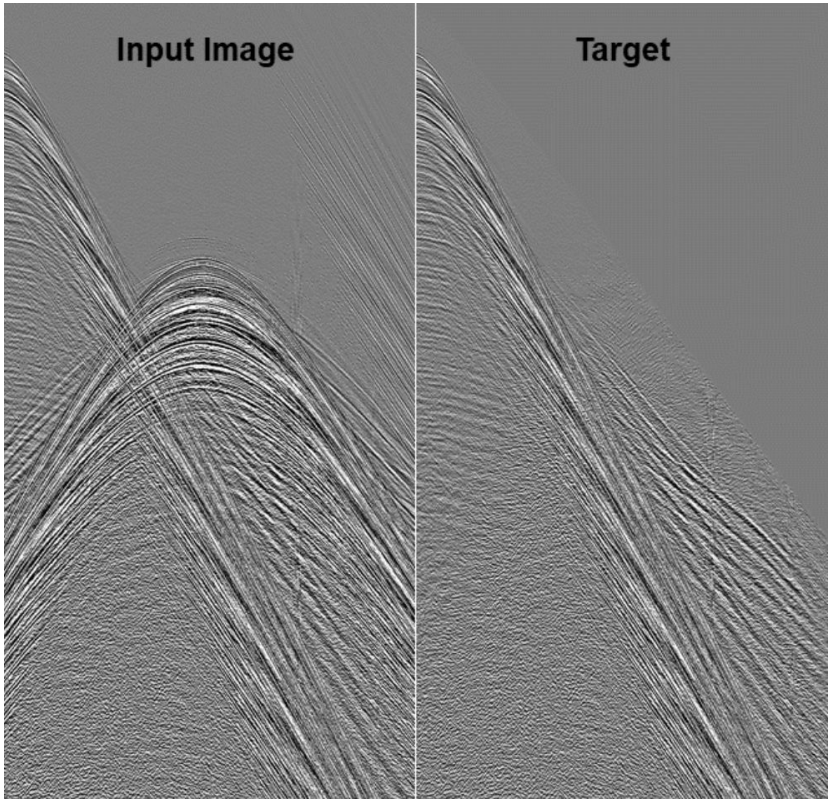
```python
class DownBlock(nn.Module):
    """
    Block of the downsampling (encoding) path.

    Parameters
    ----------
    inCha : integer
        Input channels for the convolution.
    outCha : integer
        Output channels of the convolution (number of kernels).
    pooling : bool
        Whether to perform the pooling or not at the end of the block.
    poolPad : tuple of two integers
        Padding to be used in the pooling.
    residual : bool
        Whether to use skip connetcion or not.

    Methods
    -------
    forward : forward pass into the block
    """
    def __init__(self, inCha, outCha, pooling, poolPad, residual=False):
        super(DownBlock, self).__init__()
```

```python
class UNet(nn.Module):
    """
    The network is constituted by an encoder and a decoder.
    We assume at least 2 downblocks and 1 upblock.

    Parameters
    ----------
    inCha: integer
        Number of channels in the input tensor.
    finOutChan: integer
        Number of channels in the output tensor.
    numChannels : list of integers
        Number of channels for the output of each downblock.
    inWidth : integer
        Number of columns in the input image.
    inHeight : integer
        Number of rows in the input image.
    residual : bool
        Whether to use skip connetcion or not.
    iniPad : List of integers
        Padding of the input tensor.
        Same rules as in nn.functional.pad.

    Methods
    -------
    forward : forward pass into the network
    """
    def __init__(self, inCha, finOutCha, numChannels, inWidth, inHeight, residual=False, iniPad=None):
        super(UNet, self).__init__()
```

```python
inpCha = 5
outCha = 1
lstKer = [64, 128, 256]
inpWidth = 636
inpHeight = 1251
residual = False
iniPad = [0, 0, 0, 0]

model = UNet(inpCha, outCha, lstKer, inpWidth,
             inpHeight, residual=residual, iniPad=iniPad)
```

Training the Neural Networks

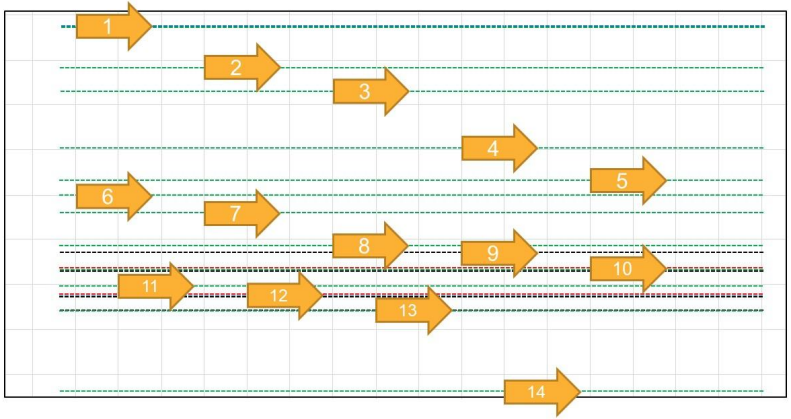
Mattia Guerri
MSc ACSE, 2018-2019



- Reference training set: data from lines 10 and 12, 2064 training examples in total.
- Validation set: data from line 11, 344 examples in total.

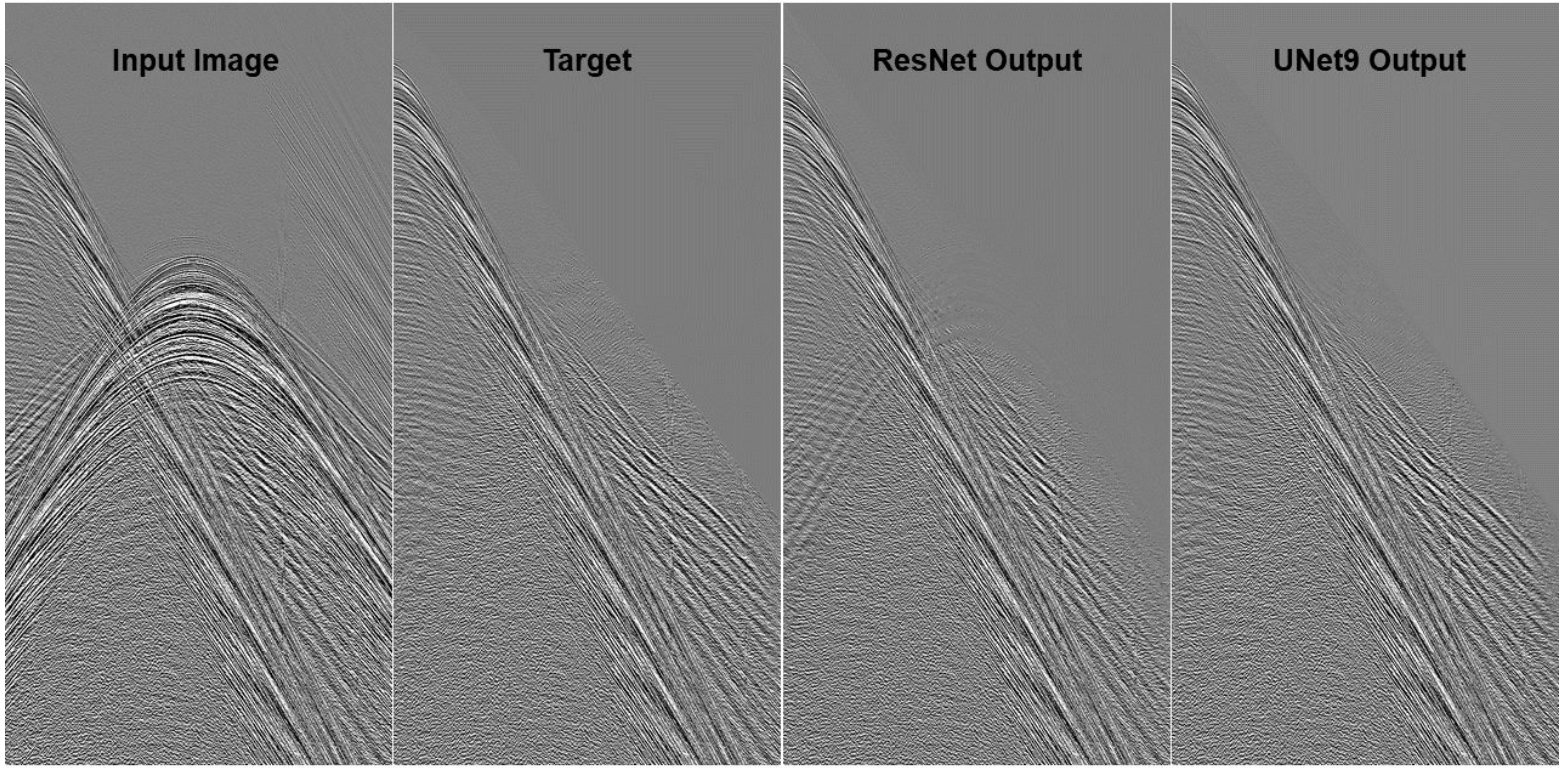
Optimisation performed with the Adam algorithm (kingma & Lei Ba, 2015) and the MSE loss function. Networks trained on 4 NVIDIA Tesla P100 or 2 Quadro RTX 6000, 12GB and 24 GB each, respectively.

Disposition of the seismic lines in a portion of the acquisition



Results

Signal Separation with Neural Networks



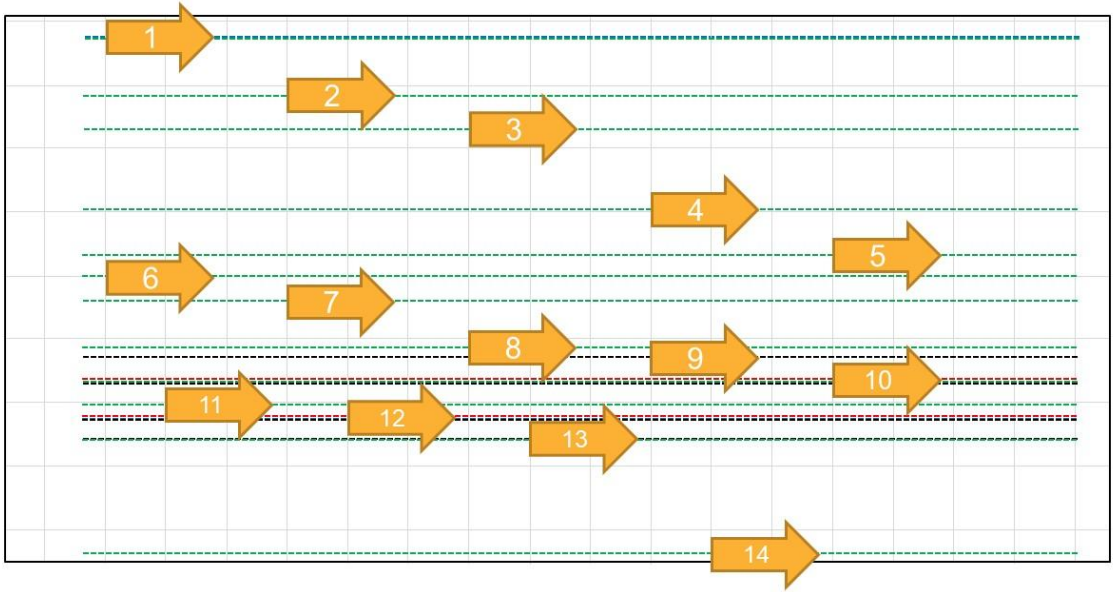
UNet9 used to de-blend signals from many lines in the acquisition. Results were QCed by CGG's production team. Exploration of UNet performance through a series of tests.

Mattia Guerri
MSc ACSE, 2018-2019

Results

Effect of Dataset Variations

- UNet9Ref trained on 10 and 12.
- UNet9_1 trained on 1 and 12.
- UNet9_2 trained on data coming from 12 lines (green).



Geological setting across the acquisition can vary significantly.

Overall, UNet9_2, trained on multiple lines across the acquisition, has a better agreement with G2.

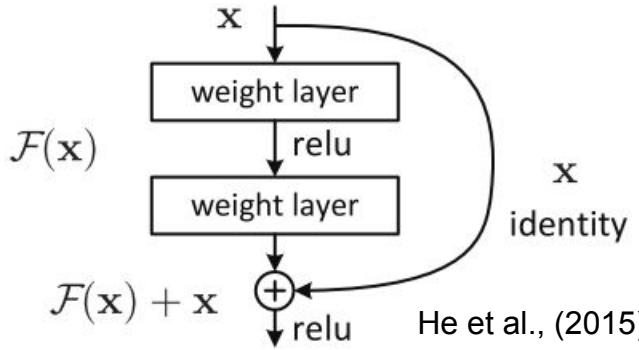
Difference of the RMS of the signals produced by G2 and the networks

Model	Acquisition Lines		
	Line 1	Line 11	Line 14
UNet9Ref	2.840	3.343	3.271
UNet9_1	2.241	3.594	3.367
UNet9_2	2.301	3.524	3.147

Effects of Architectural Modifications

Mattia Guerri
MSc ACSE, 2018-2019

- **Number of kernels increased by 50% rather than doubled, similar performance, faster training.**
- **UNet7 and UNet9 show similar performance, but UNet7 is faster to train.**
- **Residual learning helps in reducing the validation error, especially for the larger dataset.**



He et al., (2015)

Skip connection in UNet9Skip and UNet11Skip

Models trained on the reference dataset (2064) and on a dataset roughly 4 times larger.

Training on a smaller dataset strongly increases the validation loss (27.88).

Network	Reference Dataset			Dat4x		
	Train. loss	Val. loss	Train. time	Train. loss	Val. loss	Train. time
UNet7	8.795	10.40	3:37	7.684	10.04	14:04
UNet9	8.989	10.28	6:05	7.299	9.948	18:39
UNet9Half	8.958	10.90	2:10			
UNet9Skip	8.455	10.28	6:53	6.901	9.771	24:23
UNet9InPlus	8.015	10.36	11:17			
UNet9Deep3	9.232	10.42	10:51			
UNet11	8.340	10.32	10:23			
UNet11Skip	6.736	10.18	14:16	6.754	9.788	50:30

Limitations of the UNet Output

Mattia Guerri
MSc ACSE, 2018-2019

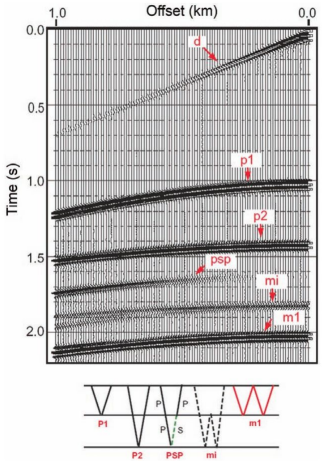
The UNet is not perfectly able to reproduce low-amplitude low-frequency features belonging to the signal that we want to preserve.

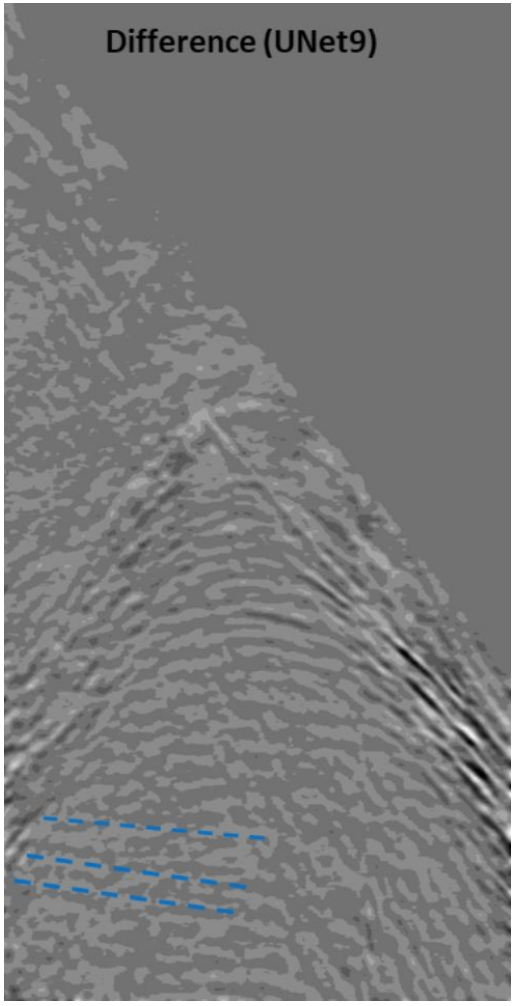
In order to tackle the issue I follow two approaches:

1. **Adding a set of convolution in front of the architecture. Large kernel size (1X101).**
2. **Using a modified loss function involving the Fourier transform of the signals in the target and network output.**

Images used in the project are the result of gathering multiple time series.

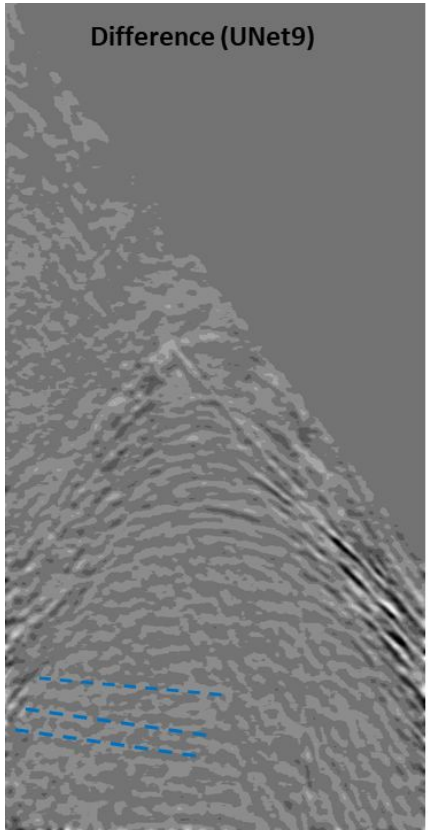
The goal is to train the network to minimise the difference in the frequency content of target and output.

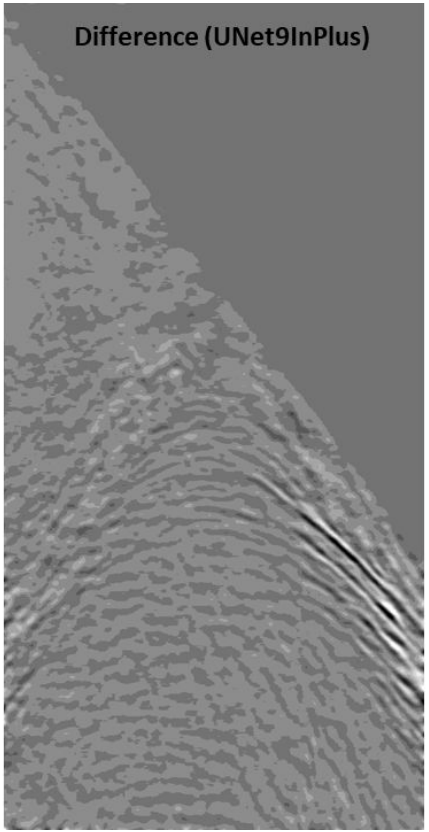


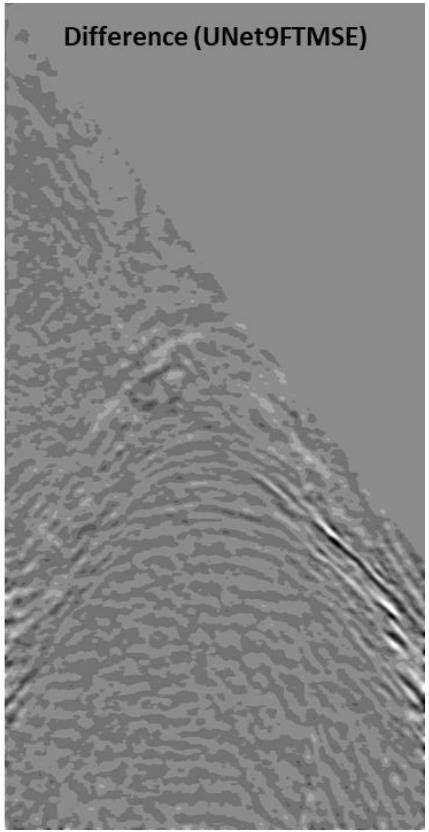


Limitations of the UNet Output

Mattia Guerri
MSc ACSE, 2018-2019







Difference between G2 and networks output. Compare to the reference model, UNet9InPlus has a training error 11% lower, UNetFTMSE 10% lower.

Project Findings

- I showed how U-shaped architectures are able to reproduce the G2 outputs, with minor limitations.
- I analysed the effect of different datasets, highlighting the importance of sampling different portions of the acquisition.
- The response of the networks to architectural modification has been investigated. Number of kernels does not have a strong impact on accuracy.
- The networks do not properly reproduce low-amplitude low-frequency signal. I proposed two approaches to tackle the issue. Both show promising results while requiring further development.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1^{st} moment vector)
$v_0 \leftarrow 0$ (Initialize 2^{nd} moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Appendix