OpenFWI: Large-scale Multi-structural Benchmark Datasets for Full Waveform Inversion
Supplemental Material

Chengyuan Deng[1,2,*] **Shihang Feng**[1,*] **Hanchen Wang**[1] **Xitong Zhang**[1,3]

Peng Jin[1,4] **Yinan Feng**[1] **Qili Zeng**[1] **Yinpeng Chen**[5] **Youzuo Lin**[1]

[1]Los Alamos National Laboratory [2]Rutgers University [3]Michigan State University
[4]The Pennsylvania State University [5]Microsoft Research
{charles.deng, shihang, hanchen.wang, xitongz, pjin, ynf, ylin}@lanl.gov
qili.zeng.cs@gmail.com, yiche@microsoft.com

Supplementary materials arrangement:

- Section 1 highlights the public resources to support the reproducibility and describes the licenses of the OpenFWI data and released code.
- Section 2 illustrates the generation pipeline of velocity maps.
- Section 3 lists the configurations of seismic forward modeling.
- Section 4 introduces the format of dataset files, the practice of naming, and other details.
- Section 5 shows the network architecture design and specifies the parameters involved.
- Section 6 provides all parameters and configurations for training to guarantee reproducibility.
- Section 7 contains more illustrations of predicted velocity maps for all datasets.
- Section 8 introduces the subsurface complexity metrics with concrete numerical results and illustrations.
- Section 9 demonstrates more details on the generalization test and analysis of the results.
- Section 10 conducts the case study of uncertainty quantification.
- Section 11 includes the robustness analysis and how to improve model robustness.
- Section 12 compares inversion results and computational cost of physics-driven methods and data-driven methods.
- Section 13 studies if predicting 2D slices by InversionNet can have comparable results as InversionNet3D.
- Section 14 elaborates the test strategy in the real-world situation.
- Section 15 establishes a connection from OpenFWI to passive geological inference and its potential applications.
- Section 16 has more discussion on previous versions of datasets and current limitations of OpenFWI.

[*]Equal contribution

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.

1 OpenFWI Public Resources and Licenses

First and foremost, the reproducibility of OpenFWI benchmarks is guaranteed by a number of public resources, listed below. Remarkably, we have a group (link available below) where any related discussion is welcome. Our team also promises to maintain the platform and support further developments based on the community feedback.

- **Website:** `https://openfwi-lanl.github.io`
- **Dataset URL:** `https://openfwi-lanl.github.io/docs/data.html#vel`
- **Github Repository:** `https://github.com/lanl/openfwi`
- **Pretrained Models:** `https://tinyurl.com/bddzkxfz`
- **Tutorial:** `https://openfwi-lanl.github.io/tutorial/`
- **Google Group:** `https://groups.google.com/g/openfwi`

The codes are released on Github under **OSS** license and **BSD-3** license, as required by the Los Alamos National Lab and the Department of Energy, U.S.A. We also attach the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License to the data.

2 Velocity Map Generation

In this section, we introduce the data generation pipelines. Essentially, the data generation follows two steps: (1) synthesizing velocity maps c and (2) generating seismic data p via forward modeling. In the first step, we generate the velocity maps from three different prior information: mathematical representations, natural images, and geological reservoir, which notably contributes to the dataset diversity. In the second step, the seismic data is obtained from the synthetic velocity maps via forward modeling. Figure 1 illustrates the data generation process with three priors elaborated below.

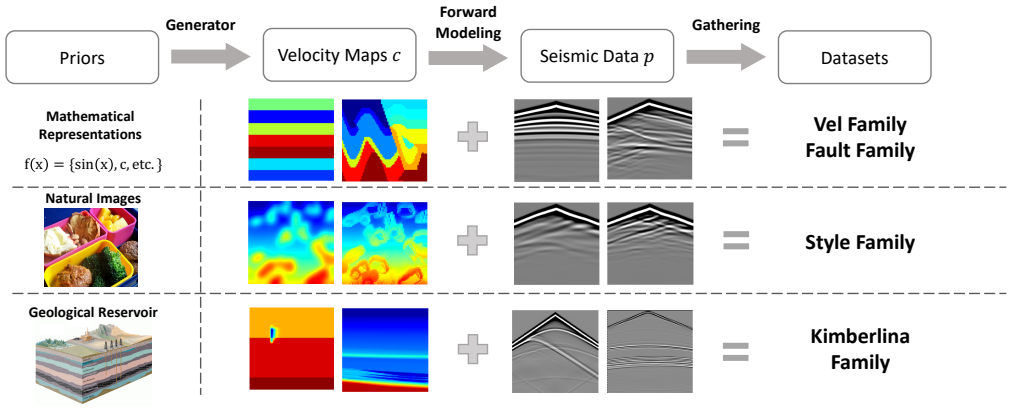

Figure 1: **Data generation pipelines.** The velocity maps c are built from three different priors, then the seismic data p are generated via forward modeling. Velocity maps c and seismic data p are collected to construct the four families of datasets.

Mathematical Representations: The sediments are usually deposited as flat-lying layers so that the velocity map c_0 is initialized as a combination of multiple flattened layers with random width. To mimic the curved folds and faults caused by the geological activities, "*Vel Family*" and "*Fault Family*" are generated by recursively applying the following equations to c_0, respectively:

$$c_i(x,y) = c_{i-1}(x, a_i \sin(2\pi k_i x));\ i > 0, \tag{1}$$

$$c_i(x,y) = \begin{cases} c_0(x + s_i, a_i \sin(2\pi k_i x) + s_i'), & y \geq f_i(x) \\ c_{i-1}(x,y), & y < f_i(x) \end{cases};\ i > 0, \tag{2}$$

where s_i, s_i', k_i and a_i are random variables in i-th iteration. $f_i(\cdot)$ is a random linear function for fault simulation. Pixels that are out of map's boundaries are filled by the nearest available values.

Table 1: **Seismic Forward Modeling Configuration**

Dataset	Grid Spacing	Source Frequency	Source Spacing	Source Numbers	Receiver Spacing	Receiver Numbers	Time Spacing	Time Steps	Boundary Grids
"Vel, Fault and Style" Family	10 m	15 Hz	140 m	5	10 m	70	0.001 s	1,001	120
Kimberlina-CO_2	10 m	10 Hz	400 m	9	40 m	101	0.002 s	1,251	120
3D Kimberlina-V1	10 m	15 Hz	800 m	5×5	100 m	40×40	0.001 s	5,001	100

Natural Images: "*Style Family*" is obtained with a style transfer network $R(\cdot)$ trained with the Marmousi model [1] as the style image and the natural images n from COCO dataset [2] as the content images[2]. The details of the network can be referred to [3]. The velocity maps are constructed using

$$c = (1 - \alpha)R(n) + \alpha c_b, \tag{3}$$

where c_b is an 1D background velocity map with linearly increasing velocity value, and α is the weight between the style-transferred velocity map $R(n)$ and c_b.

Geological Reservoir: "*Kimberlina Family*" is built under DOE's National Risk Assessment Program (NRAP) based on a potential CO_2 reservoir at the Kimberlina site in the southern San Joaquin Basin, CA, USA [4]. The hydrologic-state models h are produced with reservoir-simulation scenario [5] and then converted into velocity maps c following the petrophysical transformation methods $P(\cdot)$ [6]:

$$c = P(h). \tag{4}$$

3 Seismic Forward Modeling Details

We follow the forward modeling algorithm at `https://csim.kaust.edu.sa/files/SeismicInversion/Chapter.FD/lab.FD2.8/lab.html`. The seismic data is simulated using finite difference methods [7] with the absorbing boundary condition [8] and the Ricker wavelet is the source function. The original code in the link is written in MATLAB. To increase its computational efficiency and its compatibility with the neural network, we change its scheme to 2-4 (2nd-order accuracy in time and 4th-order in space) and rewrite it in Python. Sources and receivers are evenly distributed on the surface. The details of the forward modeling configuration are listed in Table 1, including the grid spacing of the velocity maps, the central frequency of the source wavelet, and so on. Below is an example of the acquisition geometry setting in the MATLAB script for the 1st source in "Vel, Fault and Style" datasets:

```
nz=70; nx=70;
dx=10; nbc=120; nt=1001; dt=0.001;
freq=15; s=ricker(freq,dt); isFS=false;
coord.sx = 1*dx; coord.sz = 1*dx;
coord.gx=(1:nx)*dx; coord.gz=ones(size(coord.gx))*dx;
```

4 OpenFWI Datasets: Illustration, Format, Naming, Loading

This section contains instructions on the usage of all datasets. We emphasize that all velocity maps and seismic data are saved as ".npy" files, therefore, could be conveniently accessed through Python. Other details may vary as datasets have different sizes, functions, and generation backgrounds. For the rest of this section, we go through each dataset carefully.

4.1 Vel Family

The seismic data and velocity maps are saved in two folders,`{./data}` and `{./model}`, respectively. The naming of files follows the format of `{data}{n}` for seismic data and `{model}{n}` for velocity maps, `n` denotes the index of a file (starting from 1). Notice that for the same `n`, data and model

[2]For the training, we use the Github package at https://github.com/kewellcjj/pytorch-multiple-style-transfer

become a pair. Each file contains 500 samples. The training set and testing set are split as 24K/6K, the corresponding training data are `{./data1-48.npy}` (paired with `{./model1-48.npy}`), and the rest are testing data. Examples are shown in Figure 2.

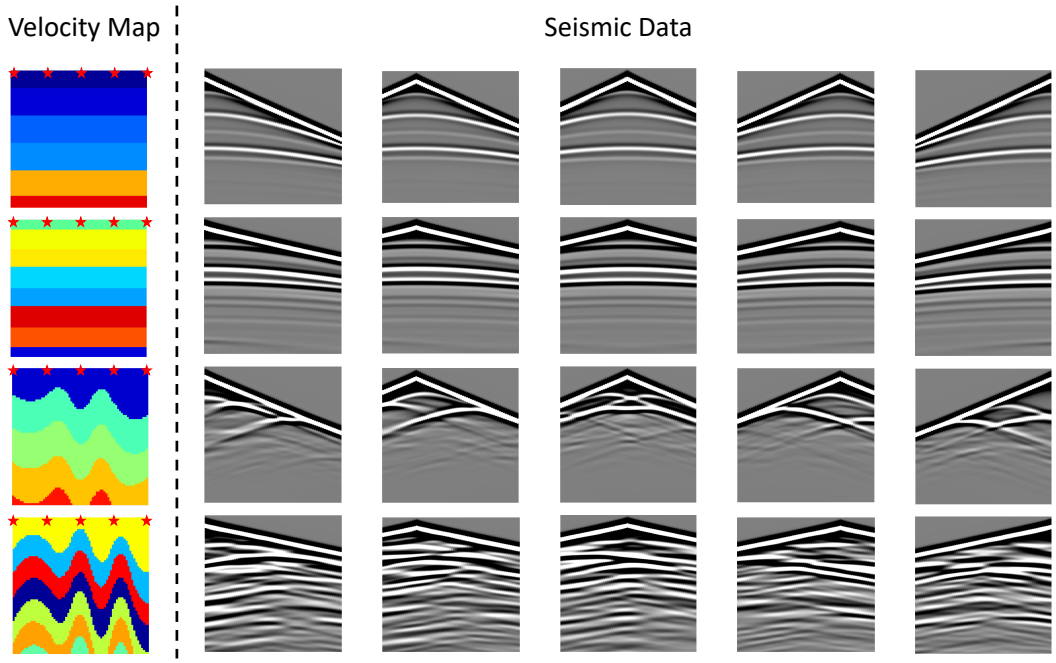

Figure 2: **Example of velocity maps and seismic data in "*Vel Family*"**. The left part shows one velocity map from each dataset, and the right demonstrates all five channels of the seismic data. The red stars in the velocity map indicate the location of sources while the receivers are distributed all over the surface.

4.2 Fault Family

The naming of files can be described as `{vel|seis}_{n}_1_{i}.npy`, where `vel` and `seis` specify if a file includes velocity maps or seismic data, `n` represents the number of initial flatten layers for velocity maps generation and `i` is the index of a file (start from 0) among the ones with the same `n`. The sizes of training and testing datasets are 48K/6K. Examples are shown in Figure 3.

4.3 Style Family

The saving and naming of "*Style Family*" is same with "*Vel Family*", seismic data `{data}{n}` are saved in `{./data}` while `{model}{n}` are saved in `{./model}`, `n` denotes the index of a file (starting from 1). Each file contains 500 samples. The training set and testing set are split as 60K/7K, the corresponding training data are `{./data1-120.npy}` (paired with `{./model1-120.npy}`), and testing data are contained in `{./data121-134.npy}` (paired with `{./model121-134.npy}`). Examples are shown in Figure 4.

4.4 Kimberlina Family

The seismic data and velocity maps of Kimberlina-CO_2 dataset are saved in four folders,`{./kimberlina_co2_`*`{phase}`*`_data}`, `{./kimberlina_co2_`*`{phase}`*`_label}`, where *{phase}* denotes "train" or "test". We use a naming convention of `{data}_sim{n}_t{m}` for seismic data and `{label}_sim{n}_t{m}` for velocity maps, `n` denotes the 4-digit index of a simulation (starting from 0), and `m` represents the timesteps from 10 to 200 (at every 10 years). Each file contains one sample, and for the same `n` and `m`, data and model become a pair. Examples of Kimberlina-CO_2 are shown in Figure 5.

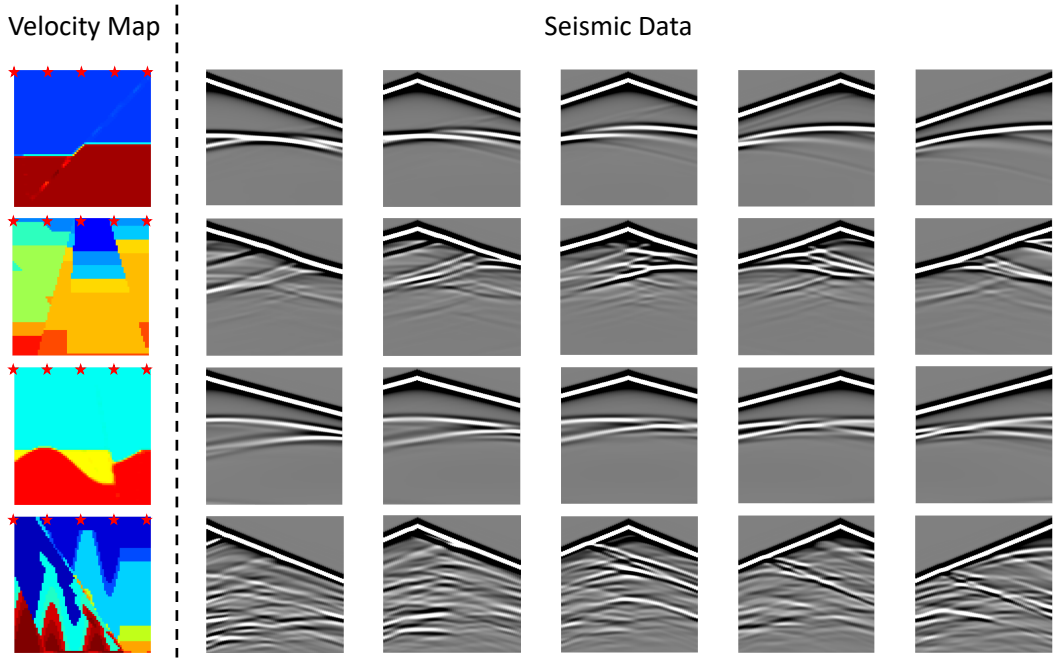

Figure 3: **Example of velocity maps and seismic data in "*Fault Family*"**. The left part shows one velocity map from each dataset, and the right demonstrates all five channels of the seismic data. The red stars in the velocity map indicate the location of sources while the receivers are distributed all over the surface.

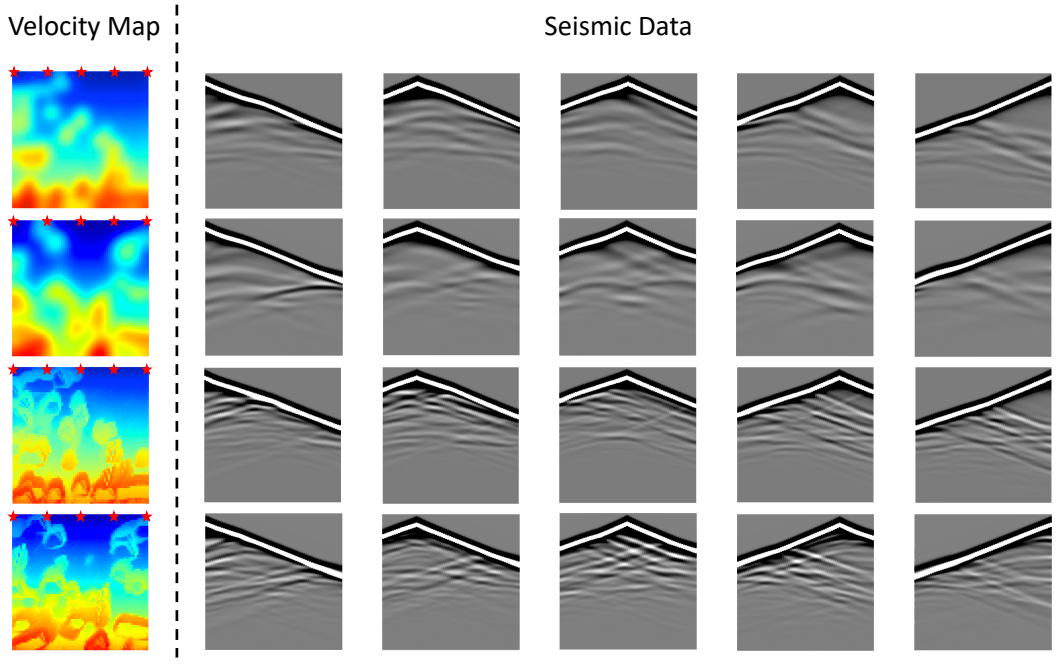

Figure 4: **Example of velocity maps and seismic data in "*Style Family*"**. The left shows the velocity maps, the first two rows are sampled from Style-A and the last two from Style-B. The right demonstrates all five channels of the seismic data. The red stars in the velocity map indicate the location of sources while the receivers are distributed all over the surface.

In 3D Kimberlina-V1[3]. The velocity maps are stored in `{./velocity_model.tar.gz}` as `{year{m}_cut{n}.npy}`. `m` represents the injection year and `n` denotes the index of a

[3]3D Kimberlina-V1 will be released at `https://edx.netl.doe.gov/` upon approval by Los Alamos National Laboratory and U.S. Department of Energy.

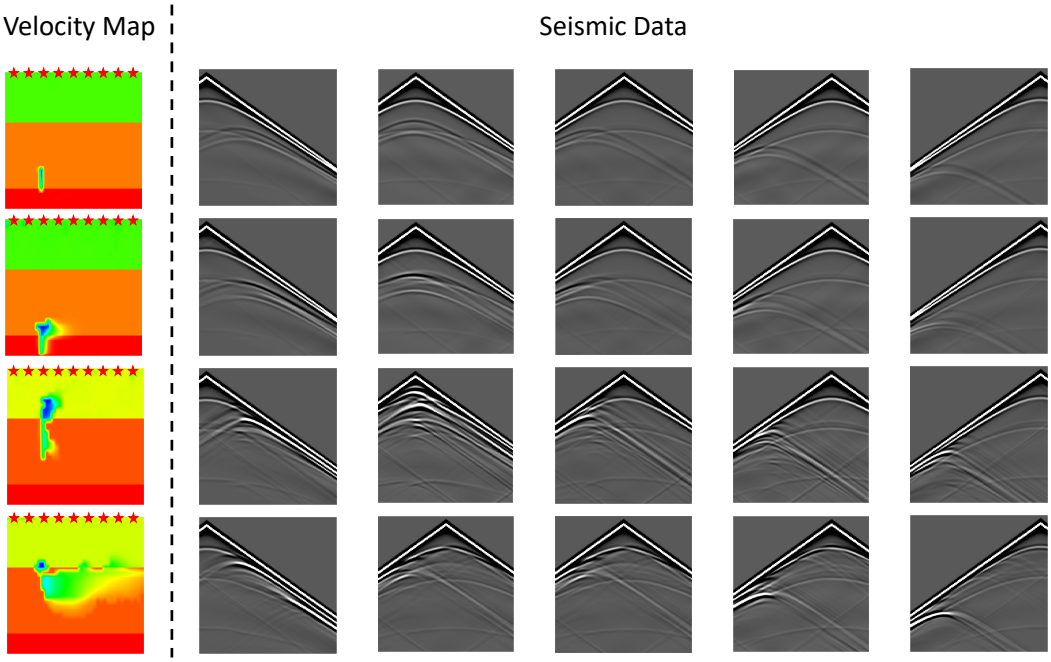

Figure 5: **Example of velocity maps and seismic data in Kimberlina-CO_2 dataset**. The left shows the velocity maps with leakage in different levels (First two rows: small; Third row: medium; Last row: Large). The right demonstrates five channels among all nine channels of the seismic data. The red stars in the velocity map indicate the location of sources while the receivers are distributed all over the surface.

file (starting from 1). Since the seismic data is too large, they are split into 12 files {`seismic_data.tar.gz.partaa`∼`al`}. Examples of 3D Kimberlina-V1 are shown in Figure 6.

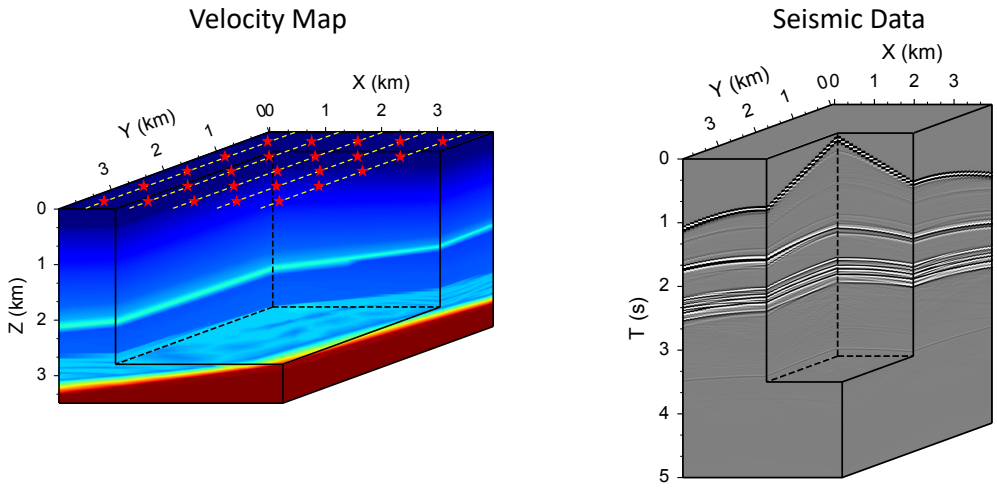

Figure 6: **Example of velocity map and seismic data in 3D Kimberlina-V1 dataset**. The left is the velocity map and the right is seismic data with the source located in the center of the surface. The red stars in the velocity map indicate the location of sources and the yellow dash lines are the survey lines. The receivers are distributed every 100 m over the surface.

5 OpenFWI Benchmarks: Network Architecture

The OpenFWI benchmarks are established upon four deep learning methods: InversionNet [9], VelocityGAN [10], and UPFWI [11] for 2D and InversionNet3D [12] for 3D FWI. All details of these methods can be retrieved from their original papers, while in this section we describe the network architecture adopted particularly for the OpenFWI datasets. Note that the ten datasets in the "*Vel*", "*Fault*" and "*Style*" family have the same size, thus share identical network architectures. The Kimberlina-CO_2 data requires a minor change on the convolution kernel parameters.

5.1 InversionNet

InversionNet is an encoder-decoder structural CNN network. The encoder extracts the hyper-features of the seismic input, and the decoder estimates the corresponding velocity map from the compressed latent vector. We stack 14 CNN layers in the encoder where the first layer has a $7 * 1$ kernel size, and the following six layers have a $3 * 1$ kernel size. Stride 2 is applied every the other layer to reduce the data dimension to the velocity map dimension. Then six $3 * 3$ CNN layers are used to extract spatial-temporal features in the compressed data, in which the data is down-sampled every the other layer using stride 2. After that, a CNN layer with an $8 * 9$ kernel size is stacked to flatten the feature maps to the output latent vector size, which is 512 in our implementation. The decoder first applies a deconvolutional layer on the latent vector to generate a $5 * 5 * 512$ tensor with a kernel size 5, followed by a convolutional layer with the same number of input and output channels. The deconvolution-convolution process is then duplicated for 4 times with a kernel size of 4 in the deconvolutional layers, resulting in a feature map with a size of $80 * 80 * 32$. Finally, we center-crop the feature map by a $70 * 70$ window and apply a $3 * 3$ convolution layer to output a single channel velocity map. Thus, there are 14 CNN layers in the encoder and 11 layers in the decoder. All the aforementioned convolutional and deconvolutional layers are followed by batch normalization and leakyReLU as the activation function.

5.2 VelocityGAN

VelocityGAN is a generative adversarial network, its generator has the same architecture as the encoder-decoder network in InversionNet. The discriminator is a 9-layer CNN. First, eight $3 * 3$ CNN layers are used to extract spatial-temporal features in the velocity maps, in which the data is down-sampled every the other layer using stride 2. Then, a CNN layer with a $5 * 5$ kernel size and zero padding is used as the output layer. Similarly, all the aforementioned convolutional layers are followed by batch normalization and leakyReLU as the activation function. To train the GAN, we use Wasserstein loss with a gradient penalty besides the pixel-wise ℓ_1-norm and ℓ_2-norm losses to distinguish the real and generated velocity maps.

5.3 UPFWI

UPFWI is an unsupervised learning method that utilizes convolutional layers in an encoder-decoder architecture. The general network architecture is the same as the one in InversionNet, except for the decoder uses CNN layers with nearest neighbor upsampling as the deconvolutional layers.

Unlike the other benchmark methods, UPFWI minimizes the data difference by the following loss function:

$$\ell(p, \tilde{p}) = \lambda_1 \ell_1(p, \tilde{p}) + \lambda_2 \ell_2(p, \tilde{p}) + \lambda_3 \ell_{per1}(\phi(p), \phi(\tilde{p})) + \lambda_4 \ell_{per2}(\phi(p), \phi(\tilde{p})), \tag{5}$$

where p and $\tilde{p}$ are the input and reconstructed seismic data from the forward operator as in section 3. ℓ_1 and ℓ_2 are the ℓ_1-norm and ℓ_2-norm losses measuring the pixel-wise data losses, respectively. And the ℓ_{per1} and ℓ_{per2} are the ℓ_1- and ℓ_2-norm distance of the perceptual losses that we extract data features from $conv3$ in a VGG-16 network pre-trained on ImageNet. $\phi(\cdot)$ represents the output features of the VGG-16 network. λ_i are the hyper-parameters that balance the four parts of the total loss.

5.4 InversionNet3D

As a natural extension of InversionNet in the 3D domain, InversionNet3D was constructed in a similar topological structure. The shallowest version, from which the results in this paper were obtained,

was built upon 13 layers in both encoder and decoder with each layer being a 3D convolution or deconvolution followed by batch normalization and LeakyReLU activation. In order to reduce memory consumption and computational complexity, two of the most important barriers of 3D FWI, the filter size and stride of each layer were deliberately chosen and another two special components, group convolution, and invertible layers were also employed in certain stages of the network. As a result, this baseline network has 14.42 million parameters and consumes 9.93GB of memory with a batch size of one. It is recommended to refer to the original paper for a more accurate and detailed description of the network architecture.

6 OpenFWI Benchmarks: Training Configurations

In this section, we provide the details of training configurations to guarantee reproducibility. All the experiments are implemented on NVIDIA Tesla P100 GPUs. We have released the pretrained model via Google drive: `https://tinyurl.com/bddzkxfz`. The codes and related information are available on Github: `https://github.com/lanl/OpenFWI`.

In 2D benchmarks, we use identical hyperparameters across all datasets for InversionNet and VelocityGAN, while the training of UPFWI varies a little on different datasets. In particular, we employ AdamW [13] optimizer with a weight decay of 1×10^{-4} and momentum parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ to update all models. For InversionNet, the learning rate is 1×10^{-4}, and no decay is applied. The size of a mini-batch is 256. We train all InversionNet models for 120 epochs. For VelocityGAN, the learning rate of both generator and discriminator is 1×10^{-4}, and no decay is applied. We set the size of a mini-batch to 64. Following the strategy of [14], we perform three discriminator iterations per generator update. All VelocityGAN models are trained for 480 epochs. For UPFWI, the initial learning rate is 3.2×10^{-4}, and we reduce the learning rate by a factor of 10 at epochs 150 and 175, except for the experiments on CurveVel-A and CurveVel-B where no decay is applied. The size of a mini-batch is 128 in the experiments on CurveVel-A and CurveVel-B and 256 in all the other experiments. We follow [11] and set the weight of each loss term to 1. Due to relatively high computational cost, we train UPFWI models for as many epochs as we can. The number of epochs in different experiments ranges from 200 to 500. In 3D benchmarks, we keep the same training configurations of InversionNet3D in all experiments, and details can be found in [12].

During training, we apply min-max normalization to rescale velocity maps and seismic data to $[-1, 1]$. The velocity values range from $1500\ m/s$ to $4500\ m/s$, and the amplitude of seismic data is between -20 and 60 in "*Vel Family*", "*Fault Family*" and "*Style Family*". For "*Kim Family*", velocity values range from $947\ m/s$ to $2545\ m/s$ in Kimberlina-CO_2 and from $1975\ m/s$ to $3892\ m/s$ in 3D Kimberlina-V1.

7 Illustration of Inversion Results

The benchmarks adopt numerical evaluations of the inversion results by SSIM. Here we present illustrations across different methods on all datasets in Figures 7 to 10. We notice that VelocityGAN usually provides the best results, but it requires more time for training. UPFWI uses the difference between predicted and observed seismic data as the loss function so that it is sensitive to the boundaries of the layers, which are the causes of the reflection waves in the data. Moreover, the deep regions of the velocity maps are hard to invert in UPFWI due to the limited seismic illuminations [15].

8 Velocity Map Complexity

In this section, we introduce three metrics applied to measure the velocity map complexity: spatial information [16], gradient sparsity index [17], and Shannon entropy [18]. Shannon entropy is the most widely-applied quantifier of "the amount of information", and by definition, it is the expectation of the logarithm of the variable in the dataset. Spatial information (SI) is an estimator of the border magnitude since it is obtained from the Sober operator [19], which is an edge detection filter. The gradient sparsity index (GSI) calculates the percentage of non-smooth pixels in an image also by applying the Sober operator. Specifically, let G_x and G_y denote the gradient magnitude on horizontal and vertical coordinates (x, y) obtained via the Sober filter, G denotes the matrix with G_p as the element on pixel p, and P denotes the total number of pixels, and then we have the following

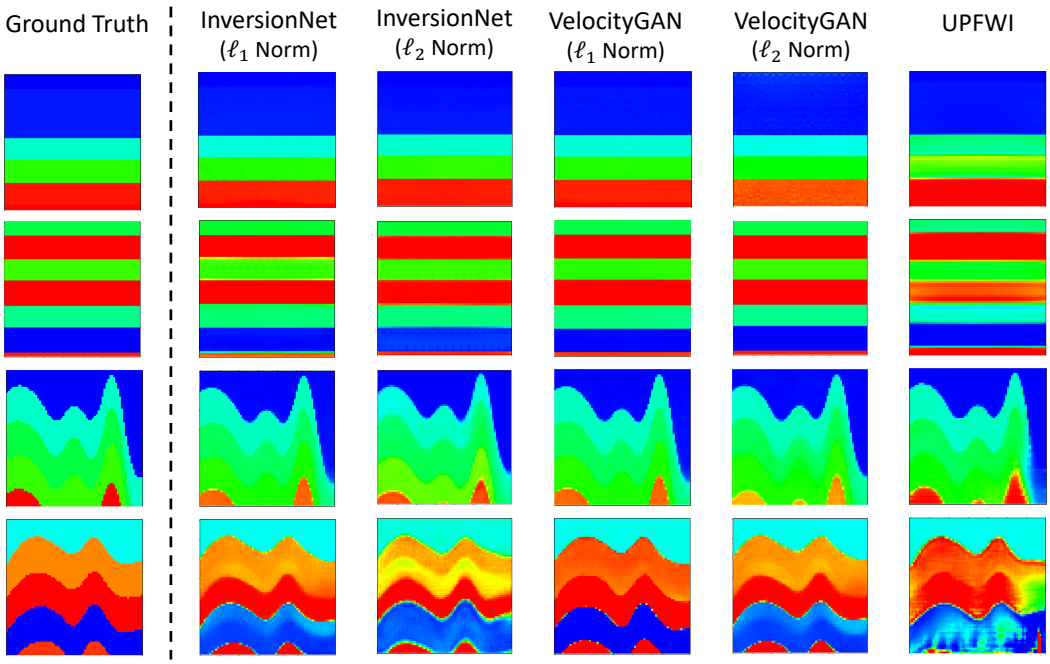

Figure 7: **Example of ground truth and inversion results in "*Vel Family*"**. The left part shows the ground truth velocity map, and the right demonstrates the inversion results with InversionNet, VelocityGAN, and UPFWI. From the first row to the last: FlatVel-A, FlatVel-B, CurveVel-A, CurveVel-B.

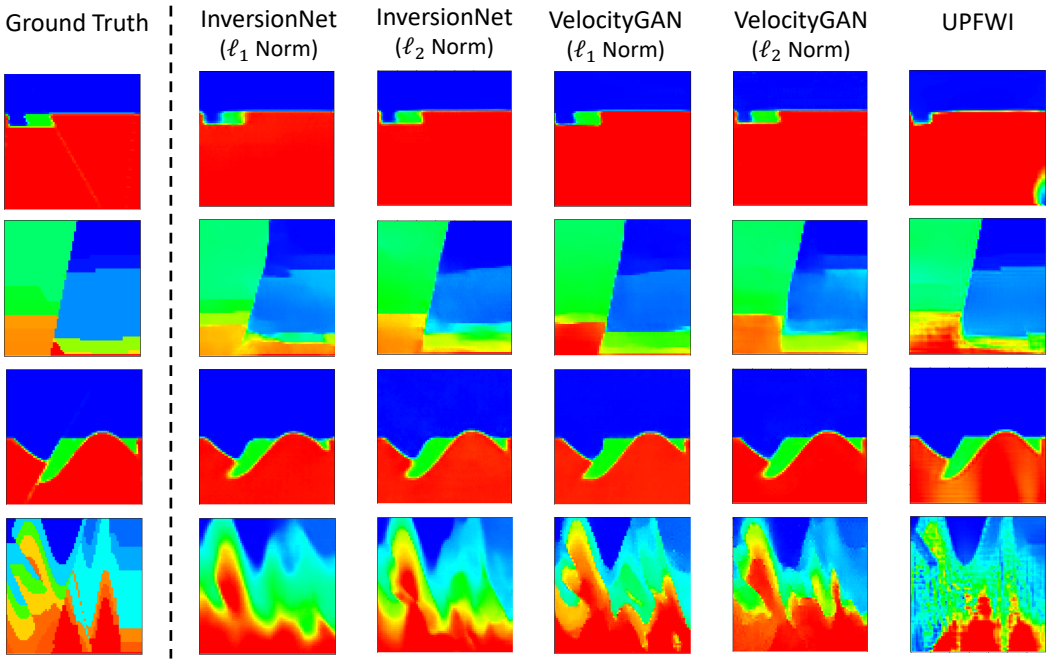

Figure 8: **Example of ground truth and inversion results in "*Fault Family*"**. The left part shows the ground truth velocity map, and the right demonstrates the inversion results with InversionNet, VelocityGAN, and UPFWI. From the first row to the last: FlatFault-A, FlatFault-B, CurveFault-A, CurveFault-B.

definitions:

$$SI_{mean} = \mathbb{E}\sqrt{(G_x^2 + G_y^2)}, \tag{6}$$

$$GSI = \frac{\|vec(G)\|_0}{P}. \tag{7}$$

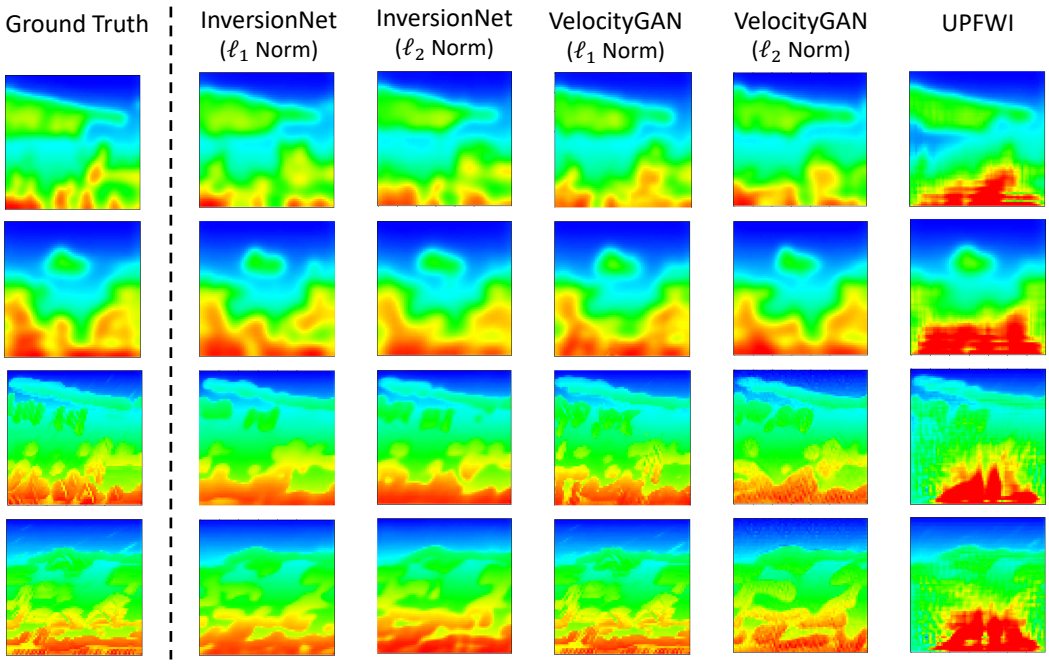

Figure 9: **Example of ground truth and inversion results in "*Style Family*"**. The left part shows the ground truth velocity map, and the right demonstrates the inversion results with InversionNet, VelocityGAN, and UPFWI. First two rows: Style-A, last two rows: Style-B.

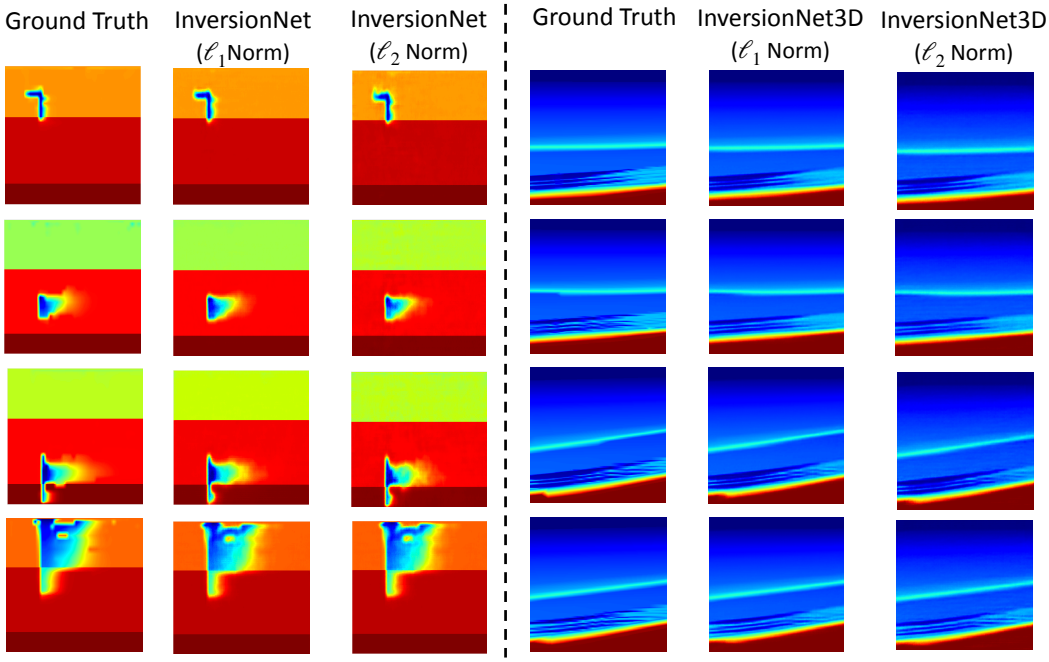

Figure 10: **Example of ground truth and inversion results in "*Kim Family*"**. The left part shows the ground truth and the InversionNet inversion results with the Kimberlina-CO_2 dataset. The right part shows the 2D slices of the ground truth and the InversionNet3D inversion results with the 3D Kimberlina-V1 dataset.

The measurement of velocity map complexity sharpens our understanding of the datasets and hopefully becomes guidance for researchers to choose appropriate datasets. The numerical results

are presented in Table 2. For more intuition, Figure 11 illustrates velocity maps with the values of three metrics of their complexity. Based on the complexity, benchmark results, and our experience, beginning users may benefit from simple datasets (FlatVel-A, FlatVel-B, CurveVel-A, FlatFault-A, Curve-Fault-A, and Kimberlina-CO_2) while advanced solutions should be evaluated on challenging datasets (CurveVel-B, FlatFault-B, CurveFault-B, Style-A, and Style-B).

Table 2: **Velocity Map Complexity** of 2D Datasets

Dataset	Spatial Information	Gradient Sparsity Index	Shannon Entropy
FlatVel-A	0.07	0.12	2.30
FlatVel-B	0.34	0.13	2.32
CurveVel-A	0.10	0.24	2.38
CurveVel-B	0.46	0.24	2.40
FlatFault-A	0.10	0.26	2.92
FlatFault-B	0.16	0.26	3.45
CurveFault-A	0.11	0.32	3.50
CurveFault-B	0.24	0.45	3.50
Style-A	0.04	1.00	12.20
Style-B	0.07	1.00	12.22
Kimberlina-CO_2	0.01	0.54	7.35

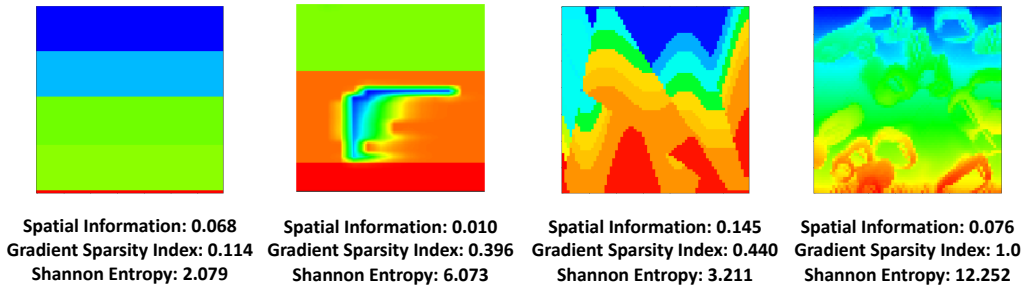

Figure 11: **Examples of different levels of velocity map complexity from each dataset family**.

We have several observations from Table 2. First, the three metrics are mostly consistent with each other for each dataset. Second, the datasets in version B always have higher complexity than version A, which meets our expectations that version B is the hard version. Third, the datasets with curve structures show higher complexity than the corresponding datasets with flat structures (e.g. CurveVel-A versus FlatVel-A).

The major limitation of this measurement is that none of the three metrics is able to provide a coherent evaluation across all datasets. For example, although "*Style Family*" datasets have more complicated details than "*Vel*" and "*Fault*" family datasets, which is consistent with the Shannon-entropy, we cannot imply it from the spatial information. We remark that the discrepancy is due to the difference in data generation methods. In short, we emphasize that although we can justify the numerical results by cross-referencing between three metrics, it would be better to have one comprehensive analysis that utilizes all three metrics.

9 Generalization Test

In this section, we provide detailed results of the generalization test on InversionNet, VelocityGAN and UPFWI in Tables 3 to 5, respectively. "FVA" is short for "FlatVel-A", and the same naming rule applies to the rest datasets. The blue, orange and green boxes indicate the *intra-domain* tests with "*Vel Family*", "*Fault Family*" and "*Style Family*", respectively. The lower triangle entries always have larger values than the upper entries for all three inversion methods, except there are a few outliers with InversionNet. The performance of VelocityGAN and UPFWI is better than the InversionNet, which is consistent with the results in [11, 10]. Moreover, InversionNet uses a larger batch size than the other two methods, which may degrade its generalization ability.

Table 3: **Generalization performance on 10 2D datasets with InversionNet**. The blue, orange and green boxes indicates the *intra-domain* tests with "*Vel Family*", "*Fault Family*" and "*Style Family*".

Source \ Target	FVA	FVB	CVA	CVB	FFA	FFB	CFA	CFB	STA	STB
FVA	–	0.51	0.55	0.30	0.84	0.52	0.66	0.43	0.51	0.50
FVB	0.42	–	0.48	0.22	0.34	0.51	0.36	0.42	0.27	0.20
CVA	0.44	0.46	–	0.25	0.39	0.54	0.42	0.42	0.39	0.29
CVB	0.69	0.53	0.52	–	0.78	0.50	0.37	0.41	0.50	0.46
FFA	0.86	0.17	0.23	0.22	–	0.23	0.46	0.23	0.30	0.34
FFB	0.42	0.45	0.50	0.19	0.40	–	0.41	0.52	0.39	0.33
CFA	0.09	0.10	0.15	0.05	0.14	0.13	–	0.14	0.56	0.51
CFB	0.38	0.40	0.50	0.19	0.38	0.59	0.42	–	0.36	0.31
STA	0.44	0.38	0.59	0.20	0.46	0.51	0.62	0.42	–	0.55
STB	0.48	0.28	0.42	0.23	0.47	0.42	0.50	0.35	0.59	–

Table 4: **Generalization performance on 10 2D datasets with VelocityGAN**. The blue, orange and green boxes indicates the *intra-domain* tests with "*Vel Family*", "*Fault Family*" and "*Style Family*".

Source \ Target	FVA	FVB	CVA	CVB	FFA	FFB	CFA	CFB	STA	STB
FVA	–	0.50	0.63	0.36	0.84	0.54	0.69	0.42	0.52	0.48
FVB	0.94	–	0.58	0.38	0.82	0.49	0.68	0.37	0.54	0.45
CVA	0.88	0.50	–	0.44	0.89	0.58	0.83	0.46	0.61	0.53
CVB	0.79	0.75	0.75	–	0.86	0.54	0.79	0.42	0.60	0.51
FFA	0.83	0.47	0.63	0.38	–	0.56	0.75	0.44	0.58	0.51
FFB	0.86	0.49	0.67	0.40	0.91	–	0.82	0.53	0.63	0.55
CFA	0.79	0.38	0.66	0.34	0.92	0.58	–	0.48	0.58	0.50
CFB	0.70	0.43	0.67	0.38	0.79	0.64	0.60	–	0.62	0.53
STA	0.66	0.37	0.56	0.32	0.67	0.53	0.63	0.43	–	0.66
STB	0.52	0.29	0.47	0.25	0.51	0.47	0.50	0.39	0.70	–

Table 5: **Generalization performance on 10 2D datasets with UPFWI**. The blue, orange and green boxes indicates the *intra-domain* tests with "*Vel Family*", "*Fault Family*" and "*Style Family*".

Source \ Target	FVA	FVB	CVA	CVB	FFA	FFB	CFA	CFB	STA	STB
FVA	–	0.49	0.63	0.28	0.84	0.54	0.70	0.41	0.58	0.53
FVB	0.64	–	0.56	0.28	0.49	0.49	0.47	0.37	0.45	0.34
CVA	0.71	0.48	–	0.38	0.68	0.55	0.69	0.45	0.52	0.43
CVB	0.70	0.57	0.67	–	0.78	0.53	0.75	0.44	0.58	0.49
FFA	0.90	0.50	0.60	0.37	–	0.53	0.72	0.42	0.54	0.49
FFB	0.66	0.50	0.61	0.34	0.62	–	0.62	0.47	0.55	0.45
CFA	0.76	0.50	0.63	0.38	0.81	0.56	–	0.46	0.60	0.50
CFB	0.52	0.34	0.48	0.29	0.53	0.43	0.52	–	0.51	0.39
STA	0.54	0.36	0.53	0.28	0.51	0.48	0.53	0.41	–	0.58
STB	0.49	0.30	0.45	0.23	0.48	0.42	0.48	0.35	0.63	–

10 Uncertainty Quantification

We further conduct experiments on CurveVel-A to quantify uncertainty in InversionNet as a case study. Following [20], we modify the network architecture by adding a dropout layer with dropout rate $p = 0.2$ after each convolutional layer except the last one. As shown in Figure 12, The uncertainty on boundaries is higher than in other regions, which implies the prediction sensitivity around the boundaries. To quantify the correlation between the uncertainty and boundaries, we calculate the Pearson correlation coefficient between the uncertainty value and the gradient magnitude on the edge. The value is 0.5462, indicating a moderate positive correlation as shown in Figure 13. As illustrated in Table 6, the uncertainty increases gradually when increasing the noise level. We also include the

Table 7: **Quantitative results of mean variance on 2D datasets.** The model is trained on CurveVel-A.

Dataset	FlatVel-A	FlatVel-B	CurveVel-A	CurveVel-B	FlatFault-A	FlatFault-B	CurveFault-A	CurveFault-B	Style-A	Style-B
Variance (10^{-3})	2.6065	4.1899	1.3903	3.5000	4.2857	2.4679	2.3639	2.3142	2.9280	2.9946

average peak-to-noise ratio (PSNR) in the table. The PSNR of a sample is defined as

$$\text{PSNR} = 10 \log_{10} \frac{(p_{max} - p_{min})^2}{\ell_2(p - p')}, \tag{8}$$

where p_{max} and p_{min} denote the maximum and minimum possible values of the seismic data in a dataset, p is the clean seismic data, and p' is the noisy data. Moreover, the uncertainty values of cross datasets are much higher than training and testing on the same dataset, which indicates that domain shifts lead to an increase in uncertainty as compared in Table 7.

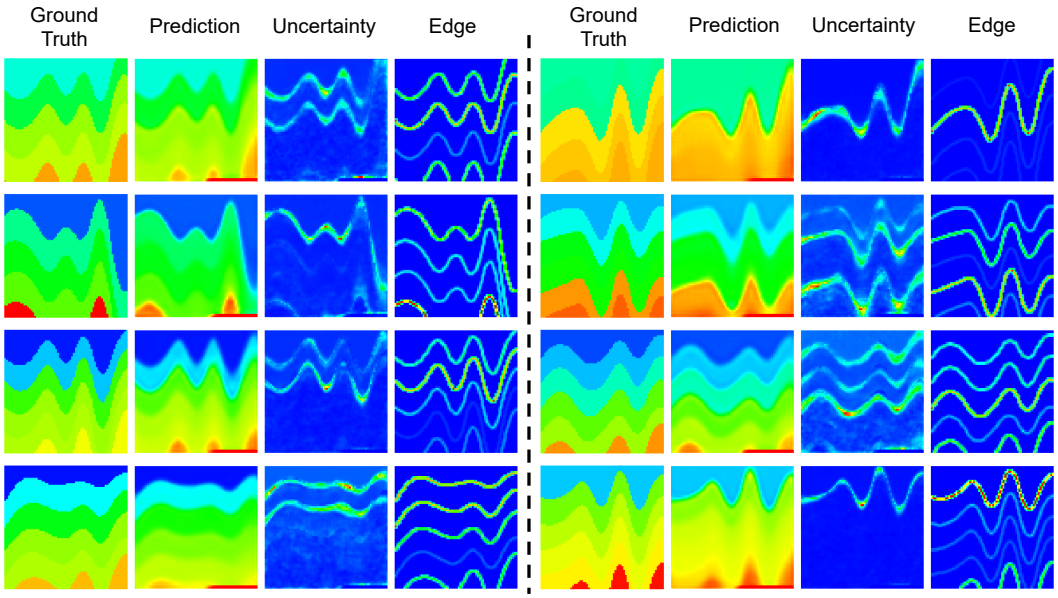

Figure 12: **Uncertainty visualization**. The uncertainty is higher on the boundaries compared with other regions.

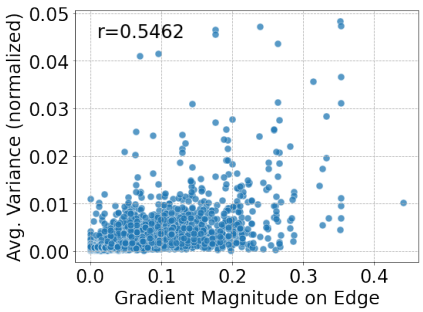

Figure 13: **The correlation between the mean variance and the gradient magnitude on velocity edge**.

Table 6: **Quantitative results of uncertainty quantification on noisy seismic input.** Gaussian noise with different standard deviation is added to seismic data during testing (σ_{test}).

σ_{test} (10^{-4})	PSNR (dB)	Variance (10^{-3})
0	100.00	1.3903
0.1	70.41	1.3970
0.5	63.40	1.4269
1.0	60.38	1.4479
5.0	53.32	1.5621

11 Robustness Test

In this section, we provide the quantitative results, visualization, and detailed analysis of the robustness test. Table 8 shows averaged performance across all 2D datasets (Kimberlina-CO_2 excluded) on both clean and noisy data. It is illustrated that InversionNets trained with ℓ_1 loss and ℓ_2 loss are both the most sensitive to noise compared to other models.

Table 8: **Quantitative results of different models on 2D and 3D datasets with noisy seismic input (Kimberlina-CO_2 excluded)**. Gaussian noise with different standard deviation σ_{test} is added to seismic data during testing. Performance in all three metrics is averaged across all datasets for 2D models. Performance is averaged across all channel selections for InversionNet3D. The least degradation is highlighted.

Method	$\sigma_{test}=0$			$\sigma_{test}=1\times10^{-5}$ PSNR=70.41dB			$\sigma_{test}=5\times10^{-5}$ PSNR=63.40dB			$\sigma_{test}=1\times10^{-4}$ PSNR=60.38dB			$\sigma_{test}=5\times10^{-4}$ PSNR=53.32dB		
	MAE↓	RMSE↓	SSIM↑	MAE↓	RMSE↓	SSIM↑	MAE↓	RMSE↓	SSIM↑	MAE↓	RMSE↓	SSIM↑	MAE↓	RMSE↓	SSIM↑
InversionNet-ℓ_1	0.0708	0.1316	0.8187	0.0769	0.1381	0.8131	0.1142	0.1879	0.7779	0.1519	0.2362	0.7419	0.2418	0.3530	0.6387
Degradation(%)	\	\	\	-8.66	-4.97	-0.69	-61.32	-42.78	-4.98	-114.70	-79.48	-9.38	-241.70	-168.26	-21.99
InversionNet-ℓ_2	0.0727	0.1211	0.8333	0.0788	0.1280	0.8290	0.1028	0.1609	0.8076	0.1348	0.2048	0.7784	0.2301	0.3309	0.6802
Degradation(%)	\	\	\	-8.45	-5.72	-0.51	-41.48	-32.91	-3.08	-85.54	-69.14	-6.59	-216.63	-173.35	-18.37
VelocityGAN-ℓ_1	0.0713	0.1286	0.8341	0.0725	0.1292	0.8329	0.0825	0.1416	0.8247	0.0951	0.1581	0.8147	0.1644	0.2526	0.7451
Degradation(%)	\	\	\	-1.73	-0.47	**-0.14**	-15.65	-10.11	-1.12	-33.35	-22.89	-2.33	-130.60	-96.35	**-10.67**
VelocityGAN-ℓ_2	0.0723	0.1209	0.8380	0.0722	0.1208	0.8364	0.0773	0.1273	0.8297	0.0858	0.1380	0.8212	0.1818	0.2722	0.7444
Degradation(%)	\	\	\	+0.11	+0.06	-0.19	**-6.91**	**-5.33**	**-0.99**	**-18.60**	**-14.18**	**-2.01**	-151.38	-125.17	-11.17
UPFWI	0.1326	0.2252	0.7716	0.1324	0.2241	0.7700	0.1437	0.2395	0.7560	0.1639	0.2680	0.7360	0.2286	0.3503	0.6725
Degradation(%)	\	\	\	**+0.14**	**+0.48**	-0.21	-8.43	-6.33	-2.01	-23.66	-18.97	-4.61	**-72.45**	**-55.51**	-12.85
InversionNet3D-ℓ_1	0.0107	0.0281	0.9837	0.0319	0.0719	0.9665	0.0320	0.0720	0.9665	0.0319	0.0719	0.9664	0.0340	0.0759	0.9731
Degradation(%)	\	\	\	-199.21	-155.41	-1.75	-199.78	-155.86	-1.75	-199.37	-155.51	-1.76	-218.59	-169.73	-1.08
InversionNet3D-ℓ_2	0.0155	0.0306	0.9462	0.0384	0.0867	0.9355	0.0388	0.0877	0.9353	0.0390	0.0879	0.9352	0.0411	0.0924	0.9346
Degradation(%)	\	\	\	-148.51	-182.93	-1.13	-151.00	-186.12	-1.15	-152.21	-186.85	-1.16	-165.80	-201.41	-1.22

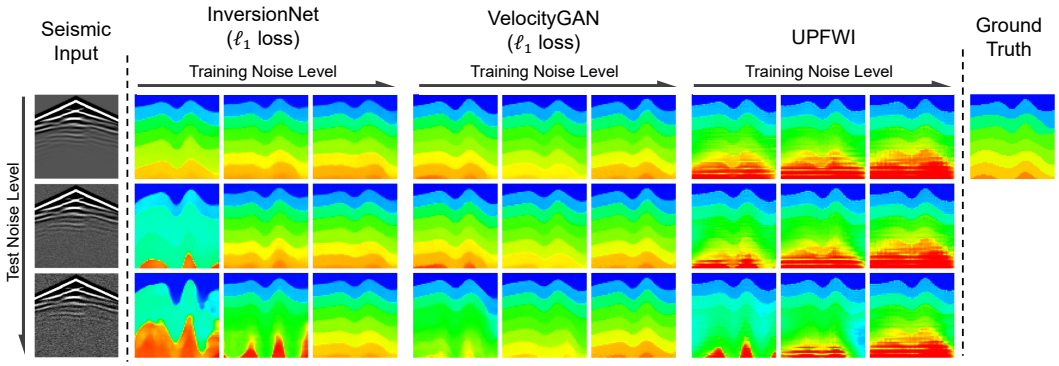

Figure 14: **Comparison of inversion results generated by different models trained with various levels of noise on CurveVel-A**. When facing noisy seismic input, the models trained with corresponding level of noise yield more accurate inversion results compared to those trained on clean data.

The results of 3D robustness test are also summarized in Table 8. We average the performance of the three-channel selections in the benchmark result. We observe that both the ℓ_1- and ℓ_2-trained InversionNet3D models are robust to Gaussian noise. The MAE/RMSE increase by 175% on average compared to clean data benchmark results, mainly due to the clean MAE/RMSE being already small. The SSIM slightly decreases by a maximum of 1.76% due to the background velocity of 3D Kimberlina-V1 remaining unchanged for all the samples which have more weight than the reservoir region when calculating SSIM.

In addition to adding noise during testing, we are also interested in whether training models on noisy data can improve robustness. We train our models on CurveVel-A with two levels of Gaussian noise ($\sigma_{train}=1\times10^{-4}$ and $\sigma_{train}=5\times10^{-4}$) added separately to seismic data. Table 9 shows quantitative results of those models on both clean data and noisy data. We observe that introducing noise into training effectively improves model robustness for all three models when the testing noise level is high. When the testing noise level is lower than the training noise level, results are mixed. Specifically, InversionNet trained with noisy data achieves better performance than the one trained on clean data, while the other two models (VelocityGAN and UPFWI) have a slight performance drop. Visualization of results is shown in Figure 14.

Table 9: **Quantitative results of different models trained with noisy seismic data on CurveVel-A.** Gaussian noise with different standard deviation is added to seismic data during training (σ_{train}) and testing (σ_{test})

Method	σ_{train} (10^{-4})	$\sigma_{test}=0$			$\sigma_{test}=1\times10^{-5}$			$\sigma_{test}=5\times10^{-5}$			$\sigma_{test}=1\times10^{-4}$			$\sigma_{test}=5\times10^{-4}$		
		MAE↓	RMSE↓	SSIM↑	MAE↓	RMSE↓	SSIM↑	MAE↓	RMSE↓	SSIM↑	MAE↓	RMSE↓	SSIM↑	MAE↓	RMSE↓	SSIM↑
InversionNet-ℓ_1	0	0.0685	0.1273	0.8074	0.0721	0.1315	0.8006	0.1530	0.2536	0.7248	0.1609	0.3600	0.6751	0.2401	0.3387	0.6081
	1	0.0675	0.1238	0.8242	0.0672	0.1236	0.8245	0.0674	0.1238	0.8246	0.0676	0.1241	0.8246	0.0780	0.1391	0.8036
	5	**0.0657**	**0.1228**	**0.8340**	**0.0659**	**0.1230**	**0.8338**	**0.0662**	**0.1234**	**0.8336**	**0.0663**	**0.1234**	**0.8338**	**0.0637**	**0.1205**	**0.8376**
InversionNet-ℓ_2	0	0.0691	0.1202	0.8223	0.0759	0.1273	0.8178	0.0830	0.1364	0.7989	0.1185	0.1839	0.7523	0.2530	0.3599	0.5887
	1	**0.0665**	**0.1165**	**0.8402**	**0.0662**	**0.1162**	**0.8405**	**0.0656**	**0.1155**	**0.8412**	**0.0655**	**0.1155**	**0.8409**	0.0894	0.1449	0.8024
	5	0.0687	0.1198	0.8360	0.0686	0.1197	0.8362	0.0683	0.1194	0.8366	0.0680	0.1189	0.8372	**0.0668**	**0.1175**	**0.8391**
VelocityGAN-ℓ_1	0	**0.0483**	**0.1034**	**0.8625**	**0.0490**	**0.1042**	**0.8615**	0.0518	0.1070	0.8583	0.0557	0.1111	0.8546	0.1331	0.2166	0.7969
	1	0.0534	0.1083	0.8630	0.0529	0.1078	0.8632	**0.0517**	**0.1065**	**0.8638**	**0.0512**	**0.1060**	**0.8639**	0.0591	0.1156	0.8534
	5	0.0524	0.1081	0.8618	0.0524	0.1081	0.8619	0.0523	0.1079	0.8621	0.0520	0.1077	0.8624	**0.0516**	**0.1073**	**0.8627**
VelocityGAN-ℓ_2	0	**0.0510**	**0.0976**	**0.8759**	**0.0519**	**0.0985**	**0.8752**	**0.0559**	**0.1027**	**0.8721**	0.0651	0.1136	0.8652	0.2025	0.3103	0.7643
	1	0.0627	0.1122	0.8333	0.0622	0.1117	0.8333	0.0608	0.1104	0.8335	**0.0603**	**0.1098**	**0.8333**	0.0666	0.1170	0.8258
	5	0.0618	0.1072	0.8083	0.0617	0.1070	0.8084	0.0613	0.1065	0.8089	0.0610	0.1061	0.8093	**0.0606**	**0.1054**	**0.8100**
UPFWI	0	**0.0805**	**0.1411**	**0.8443**	**0.0798**	**0.1411**	**0.8437**	**0.0835**	**0.1481**	**0.8379**	0.0927	0.1620	0.8292	0.1897	0.2988	0.7683
	1	0.0869	0.1604	0.8278	0.0870	0.1608	0.8279	0.0877	0.1632	0.8280	**0.0894**	**0.1676**	**0.8272**	0.1206	0.2241	0.8057
	5	0.0923	0.1671	0.8233	0.0922	0.1671	0.8233	0.0921	0.1673	0.8233	0.0920	0.1676	0.8233	**0.0953**	**0.1785**	**0.8188**

12 Comparison with Physics-driven Methods

In this section, we select 500 samples from each 2D dataset and 1 samples from the 3D dataset to test the performance of physics-driven methods. Multi-scale full waveform inversion [21, 22] is performed with the six different low-pass filters that the cutoff frequencies equal to $(1, 3, 5, 10, 20, 30)$ Hz. We use three different initial maps in the inversion: homogeneous maps, linear increasing maps, and smoothed maps. Homogeneous maps are built with a constant velocity equal to minimum velocity on the surface. The velocity increases linearly with depth in linear increasing maps. Smoothed maps are obtained by applying mean filters to the ground truth velocity maps. ℓ_2-norm loss function and the conjugated gradient method are used for the iterative updating of the velocity maps. The inversion stops when the loss change is less than 0.1%.

Quantitative results of the physics-driven methods are given in Table 10 and their illustrations are presented in Figures 15 to 18. Compared with data-driven FWI results in Figures 7 to 10, physics-driven FWI results have more artifacts and the performance of physics-driven FWI strongly depends on the quality of initial maps. The initial map with higher SSIM leads to FWI results with higher SSIM. The building of the initial maps is very important and how to obtain a good initial map is still an open research topic [23, 24, 25]. For 3D Kimberlina-V1 data, the loss is hard to converge with homogeneous and linear increasing initial maps due to their poor qualities. The inversion with smoothed initial maps is greatly affected by the acquisition footprint [26]. As a result, the inversion results have strong artifacts at the surface in Figure 18 and their SSIM is even lower than the initial maps in Table 10.

Computational cost comparison: The per-sample computation time of physics-driven methods and the average training time of data-driven methods for each sample is given in Table 11. Recall that all the experiments are implemented on NVIDIA Tesla P100 GPUs. Physic-driven methods and data-driven methods measure computational cost differently. Physics-driven methods do not require a training stage but directly optimize the testing data for multiple epochs to generate the inversion results, while data-driven methods need to be trained for several epochs and then applied to the test data for a one-time inference. Thus, the computational comparison varies over different ratios between the number of training and test samples. The relationship between total computation time per test sample and train/test ratio is given in Figure 19. The total computation time of data-driven methods is the summation of the training and test time. The physics-driven methods do not need training, so the total computation time is their computation time on a test set only.

All three data-driven methods are faster when the ratio between the number of training and test samples is less than 62. Typically, the ratios for all twelve OpenFWI datasets are significantly lower (about 10) as in Figure 20. Therefore, data-driven methods speed up 6 times more than physics-driven methods.

The time complexity of physics-driven methods is $O(N^3)$ for 2D cases and $O(N^4)$ for 3D cases. Their computational costs grow much faster than those of data-driven methods when the sizes of the velocity maps and seismic data increase. For example, the total computation time of the physics-driven method with 3D Kimberlina-V1 dataset is 475 times more than that of Vel Family. For InversionNet with 3D Kimberlina-V1 dataset, the total computation time is only 40 times more than that of Vel Family. In conclusion, data-driven methods have obvious computational advantages on large velocity maps.

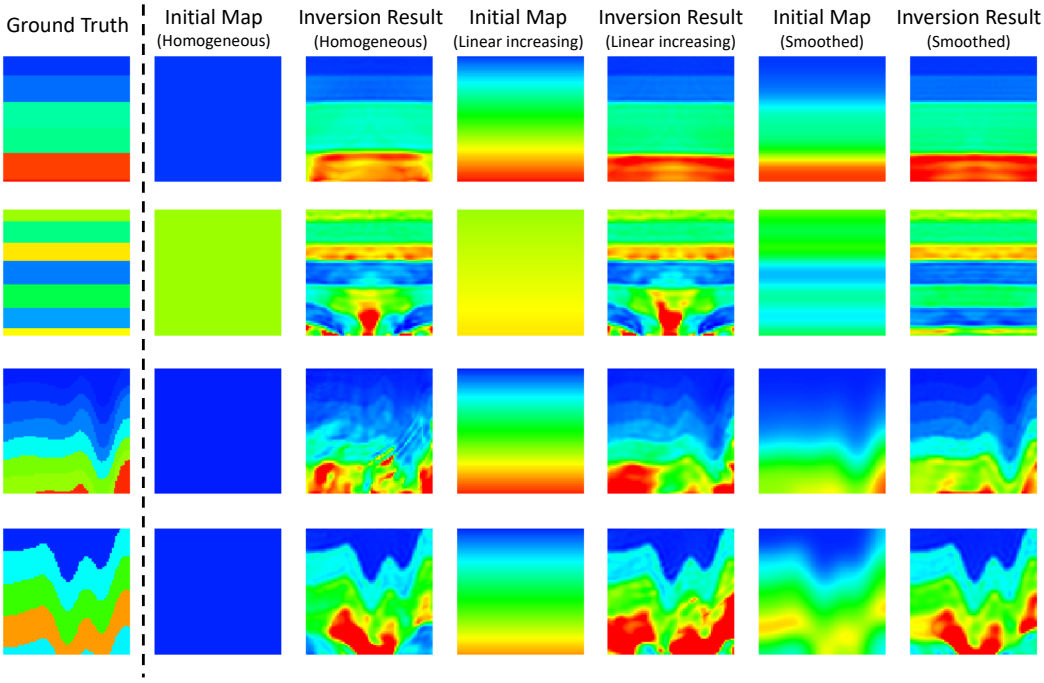

Figure 15: **Example of ground truth and physics-driven inversion results in "*Vel Family*"**. The left part shows the ground truth velocity map, and the right demonstrates the initial maps and inversion results with homogeneous, linear increasing, and smoothed initial maps. From the first row to the last: FlatVel-A, FlatVel-B, CurveVel-A, CurveVel-B.

13 Comparison between InversionNet and InversionNet3D

In this section, we compare the performance of the InversionNet when dealing with 3D data as 2D slices with the benchmark performance of InversionNet3D. Treating the Earth as 2D planes is a common way to deal with real data. To quantify the difference between 2D and 3D training strategies, we perform the following two 2D experiments and compare their results with the InversionNet3D benchmark results on the 3D Kimberlina-V1 dataset.

In the first experiment (3D simulation, 2D slices), we slice each "velocity/seismic" data sample in the 3D Kimberlina-V1 dataset into five vertical 2D slices along both inline (X) and cross-line (Y) directions according to the source locations. Thus, each 3D sample generates 10 2D velocity/seismic data pairs as training/validation samples. In the second experiment (2D simulation), we slice each "velocity" sample in the 3D Kimberlina-V1 dataset into 40 vertical 2D slices along the inline (X) direction according to the receiver locations. We then use the benchmark 2D finite difference forward operator to generate seismic data in the 2D planes and pair it with the corresponding 2D velocity slices as training/validation samples. The sources and receivers are in the same 2D vertical planes. In total, 18K 2D slice pairs are generated in the first experiment and 73K are generated in the second experiment. The samples are used to train an InversionNet under a $90/10$ training/validation separation ratio. Once trained, we test the networks' performance and compare the results with the InversionNet3D benchmark. The performance comparison is listed in Table 12. The table shows that the InversionNet performance is comparable to the InversionNet3D performance. For the 3D simulation 2D slices experiment, the MAE loss and the RMSE loss of InversionNet are about 20% less than the ones of InversionNet3D using an ℓ_1 training loss, and are about 50% larger when using

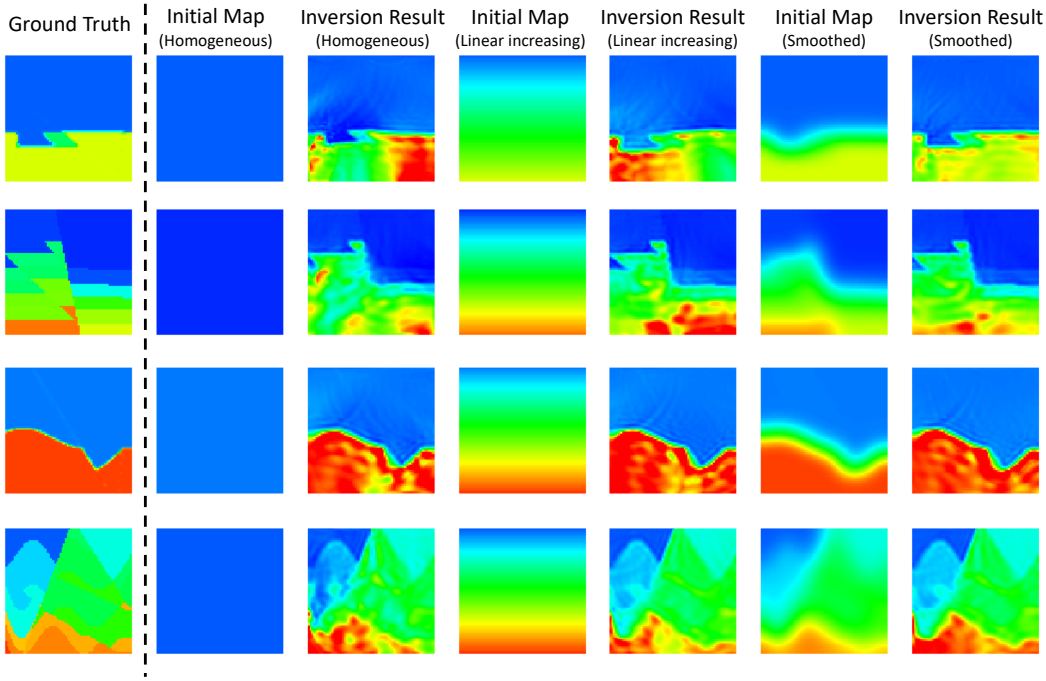

Figure 16: **Example of ground truth and physics-driven inversion results in "*Fault Family*"**. The left part shows the ground truth velocity map, and the right demonstrates the initial maps and inversion results with homogeneous, linear increasing and smoothed initial maps. From the first row to the last: FlatFault-A, FlatFault-B, CurveFault-A, CurveFault-B.

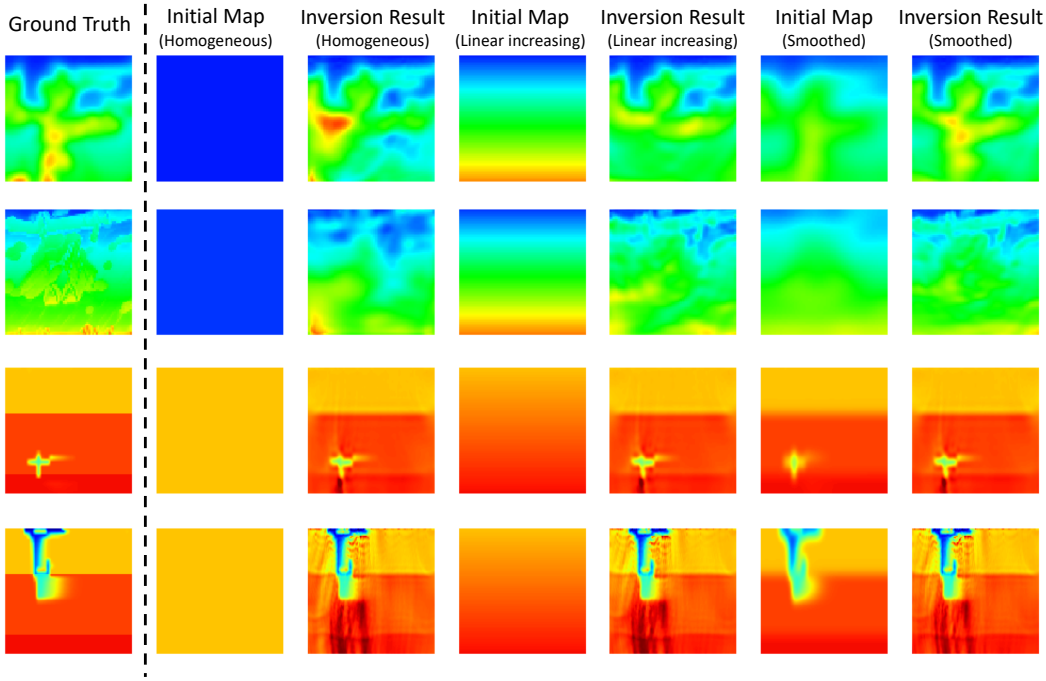

Figure 17: **Example of ground truth and physics-driven inversion results in "*Style Family*"**. The left part shows the ground truth velocity map, and the right demonstrates the initial maps and inversion results with homogeneous, linear increasing, and smoothed initial maps. First two rows: Style-A and Style-B, last two rows: Kimberlina-CO_2.

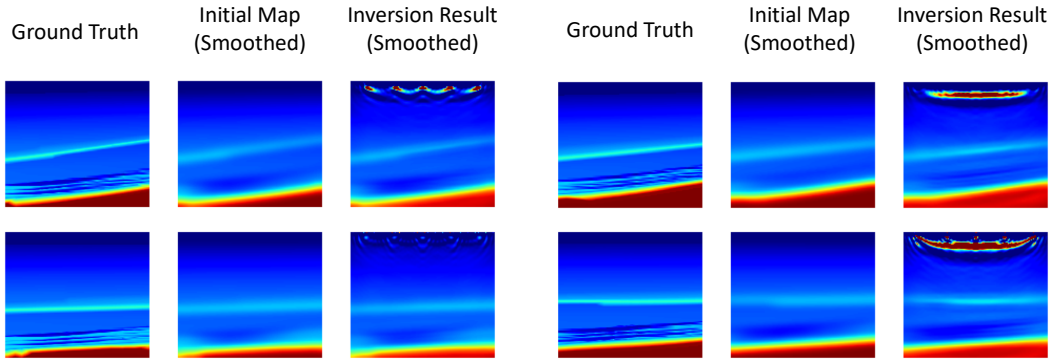

Figure 18: **Example of ground truth and physics-driven inversion results in *"3D Kimberlina-V1"***. The ground truth velocity map, the smoothed initial maps and inversion results.

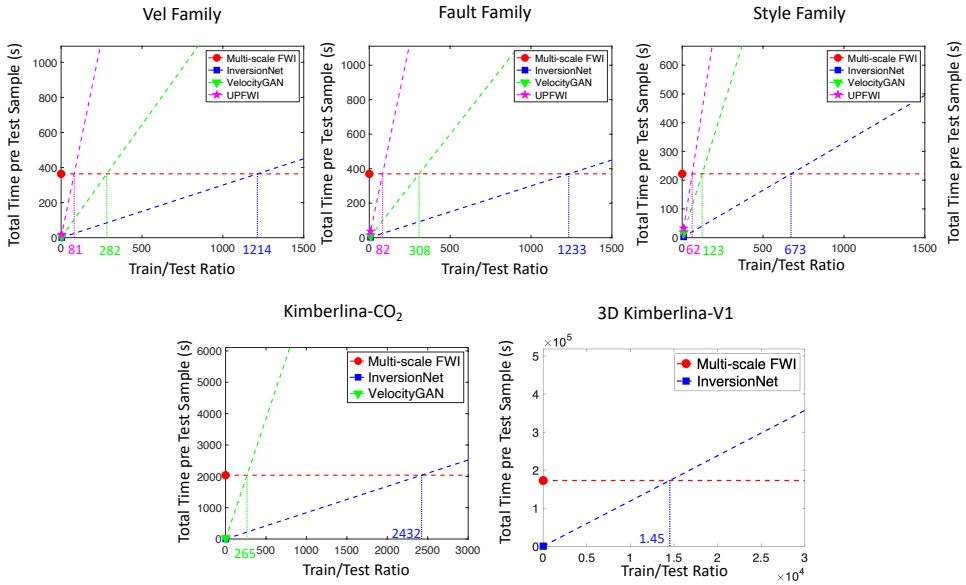

Figure 19: **The relationship between total computation time per test sample and train/test ratio**. The total computation time of data-driven methods is the summation of the training and test time. The purple, green and blue numbers are the train/test ratio when the computation time of the data-driven methods (UPFWI, VelocityGAN and InversionNet) is equal to the physics-driven method (Multi-scale FWI).

an ℓ_2 training loss. The SSIM of InversionNet is 1.86% lower than the one of InversionNet3D using an ℓ_1 training loss, and is about 0.38% higher when using an ℓ_2 training loss. For the 2D simulation experiment, the MAE loss and the RMSE loss of InversionNet are about 68% less than the ones of InversionNet3D using an ℓ_1 training loss, and are about 87% smaller when using an ℓ_2 training loss. The SSIM of InversionNet is 0.79% higher than the one of InversionNet3D using an ℓ_1 training loss, and is about 4.55% higher when using an ℓ_2 training loss.

To conclude, the performance of an end-to-end InversionNet that assumes the 3D Earth as 2D planes is comparable to that of the InversionNet3D. However, 3D training returns the whole volume of the region while 2D training returns only a few slices of the 3D volume. People may need further processing, for example, interpolation between slices, to obtain the whole 3D velocity volume.

Table 10: **Quantitative results** of physics-driven FWI on OpenFWI datasets with homogeneous, linear increasing, smoothed maps as initial maps.

Dataset	Map	Homogeneous Map			Linear Increasing Map			Smoothed Map		
		MAE↓	RMSE↓	SSIM↑	MAE↓	RMSE↓	SSIM↑	MAE↓	RMSE↓	SSIM↑
FlatVel-A	Initial	0.5553	0.7536	0.2249	0.1841	0.2438	0.4657	0.0641	0.1089	0.7087
	Result	0.0643	0.1318	**0.8009**	0.0426	0.0768	**0.8370**	0.0324	0.0621	**0.8689**
FlatVel-B	Initial	0.5222	0.7186	0.1568	0.4899	0.6752	0.1547	0.1752	0.2634	0.4850
	Result	0.1920	0.3712	**0.5495**	0.2112	0.4042	**0.5293**	0.0938	0.1902	**0.6908**
CurveVel-A	Initial	0.5599	0.7578	0.1968	0.2397	0.3178	0.3415	0.0871	0.1405	0.6349
	Result	0.1173	0.2120	**0.6447**	0.0966	0.1666	**0.6654**	0.0721	0.1282	**0.7347**
CurveVel-B	Initial	0.5129	0.7199	0.1528	0.4901	0.6699	0.1395	0.2166	0.3150	0.3991
	Result	0.2465	0.4084	**0.4157**	0.2670	0.4402	**0.3966**	0.1734	0.3020	**0.5142**
FlatFault-A	Initial	0.5123	0.7737	0.1224	0.2728	0.3613	0.3309	0.0619	0.1543	0.7846
	Result	0.0559	0.1102	**0.8222**	0.0560	0.1095	**0.8244**	0.0412	0.0834	**0.8607**
FlatFault-B	Initial	0.8476	1.0324	0.0028	0.2881	0.3682	0.2819	0.1082	0.1705	0.5790
	Result	0.1442	0.2366	**0.5547**	0.1247	0.1992	**0.5907**	0.0961	0.1599	**0.6596**
CurveFault-A	Initial	0.5064	0.7567	0.1288	0.2730	0.3604	0.3381	0.0655	0.1533	0.7886
	Result	0.0764	0.1397	**0.7652**	0.0742	0.1369	**0.7716**	0.0563	0.1078	0.8190
CurveFault-B	Initial	0.8493	1.0297	0.0063	0.3084	0.3931	0.2186	0.1484	0.2191	0.4672
	Result	0.1783	0.2832	**0.4837**	0.1637	0.2537	**0.5078**	0.1311	0.2066	**0.5765**
Style-A	Initial	0.7571	0.8812	0.1395	0.2513	0.3195	0.2881	0.0775	0.1042	0.6703
	Result	0.1229	0.2007	**0.6018**	0.0903	0.1491	**0.6866**	0.0552	0.0948	**0.7986**
Style-B	Initial	0.8165	0.9219	0.0710	0.1564	0.1927	0.3578	0.0910	0.1080	0.4931
	Result	0.1344	0.1914	**0.4599**	0.0764	0.1102	**0.6521**	0.0821	0.1168	**0.6132**
Kimberlina-CO_2	Initial	0.2033	0.2656	0.8198	0.0792	0.1512	0.8873	0.0689	0.1061	0.8924
	Result	0.0434	0.0914	**0.9137**	0.0419	0.0904	**0.9125**	0.0404	0.0896	**0.9122**
3D Kimberlina-V1	Initial	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	0.1174	0.2891	**0.7975**
	Result	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	0.5248	0.7581	0.5861

Table 11: **Pre sample computation time of physics-driven and data-driven FWI**.

Computation Time	Method	Vel Family	Fault Family	Style Family	Kimberlina-CO_2	3D Kimberlina-V1
Physics-driven FWI t_p	Multi-scale FWI	364s	370s	222s	2036s	48h
Data-driven FWI t_d	InversionNet	0.3s	0.3s	0.33s	0.84s	11.90s
	VelocityGAN	1.29s	1.2s	1.8s	7.68s	N.A.
	UPFWI	4.5s	4.5s	3.6s	N.A.	N.A.

Table 12: **Quantitative results** of two InversionNet training strategies compared with the averaged InversionNet3D benchmark on the 3D Kimberlina-V1 dataset.

Network Architecture	Data Generator	MAE↓	RMSE↓	SSIM↑
InversionNet-ℓ_1	3D simulation, 2D slices	0.0080	0.0215	0.9652
	2D simulation	0.0028	0.0093	0.9917
InversionNet-ℓ_2	3D simulation, 2D slices	0.0300	0.0401	0.9500
	2D simulation	0.0019	0.0042	0.9974
InversionNet3D-ℓ_1	3D simulation	0.0105	0.0276	0.9838
InversionNet3D-ℓ_2	3D simulation	0.0155	0.0306	0.9462

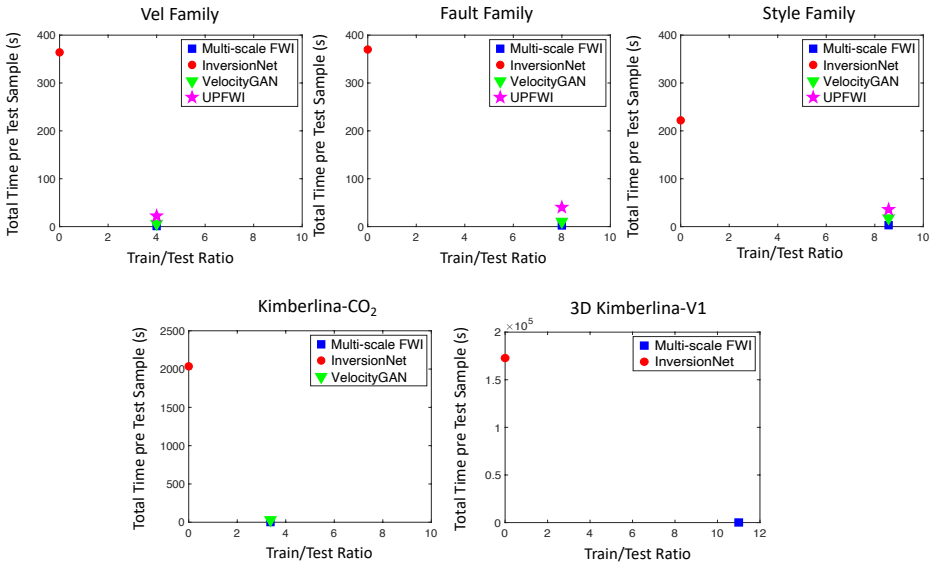

Figure 20: **The comparison between total computation time per test sample of physics-driven methods and data-driven methods in OpenFWI regarding to train/test ratio**. The computation time of data-driven methods (square, triangle and star markers) is much loss than the physics-driven method (circle marker) in OpenFWI dataset.

14 Test Strategy in Real-world Situation

The selection of the best inversion model is to minimize the discrepancy between the predicted velocity map and ground truth, which can be written as

$$\arg\min_{m} D(c_{pred}, c_{true}) \tag{9}$$

where D is the discrepancy, c_{pred} is the predicted velocity map given by inversion model m and c_{true} is the ground truth. The discrepancy D may be obtained by calculation (ℓ_1 or ℓ_2 norm), or may be obtained from domain experts.

In the real-world case, c_{true} may not be available . If we have a prior velocity map c_{prior}, we will be able to calculate $\arg\min_{m} D(c_{pred}, c_{prior})$ to approximate Eq (9). However, the inversion models must be trained to give c_{pred}. In order to save computational cost, we can minimize the discrepancy between the training set and the prior velocity map:

$$\arg\min_{m} \sum_{train_i} D(c_{train_i}, c_{prior}) \tag{10}$$

where $train_i$ is the index of the training samples corresponding to m.

If there is no prior information, we would suggest using the seismic loss as the discrepancy, which is similar to physics-driven FWI:

$$\arg\min_{m} ||f(c_{pred}) - d_{true}||^2 \tag{11}$$

where f is the forward modeling operator and d_{true} is the observed seismic data. Below is the demonstration of this scheme.

Here we conduct an experiment to demonstrate how to choose a dataset for the target in the real scenario. We choose a real velocity map in the Gulf of Mexico [27] and simulate the seismic data as the pseudo-real data. Then we apply all twenty models trained across ten datasets (two models trained

per dataset using ℓ_1 and ℓ_2 norm, respectively). We choose the two models with the minimal seismic loss. The RMSE between the predicted seismic data and the pseudo-real data is given in Figure 21. The best model trained using ℓ_1 is from the FlatVel-B dataset and the best model trained using ℓ_2 is from the Style-A dataset. Both models generate reasonably good velocity maps.

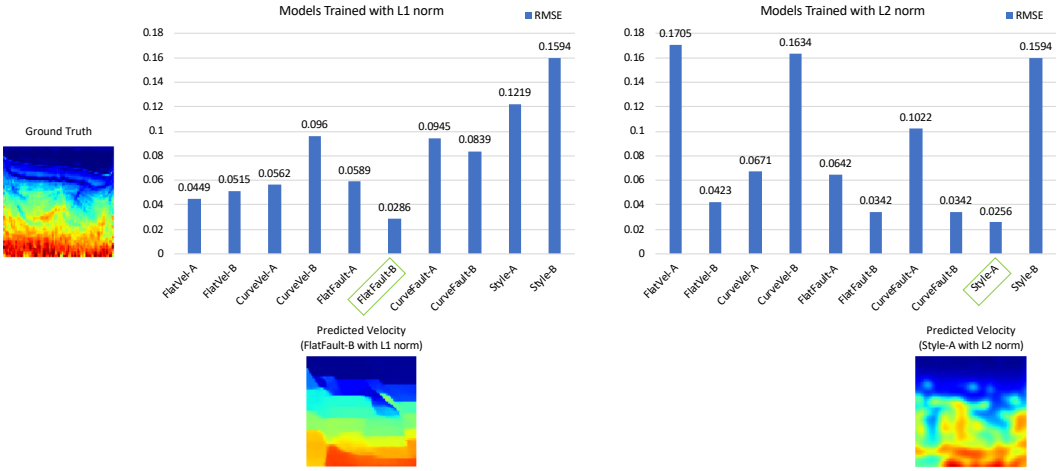

Figure 21: **Seismic data loss between predicted seismic data and pseudo-real data**. Among all the models trained with $\ell 1$ norm and $\ell 2$ norm, models trained with FlatFault-B and Style-A have the lowest RMSE, respectively. The predicted velocity maps with these two models are shown.

All of the methods mentioned above would require some additional effort to select the best datasets to use. If we would like to choose one with minimal effort, we would suggest using the "Style Family" in that this particular dataset yields highly diversified features obtained from natural images. That would enable the inversion of field data in general cases. You may choose version A or version B depending on the requirement of the velocity resolution.

15 Passive Picking

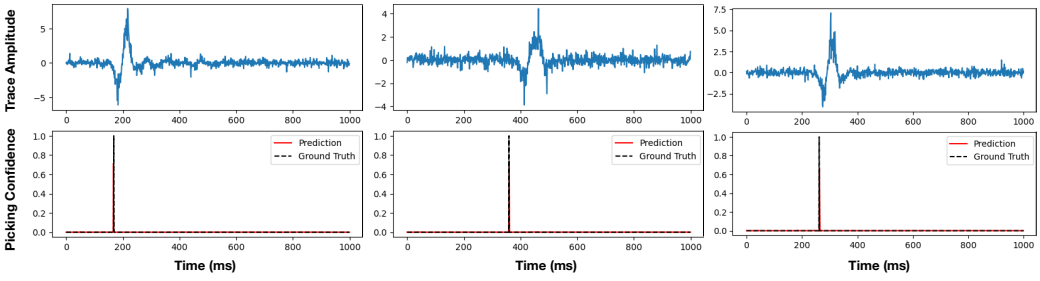

Figure 22: **P-wave arrival pickings** by InversionNet trained on the Style-B dataset. Arrival labels manually generated and used in the training stage.

The current OpenFWI datasets mainly focus on geophysical imaging problems with active or controlled seismic sources, meaning the seismic sources are usually controlled explosions or generated by vibrators. However, there is another sub-field of geophysical imaging problems, passive seismic imaging, in which the source term is unknown. It is a common problem in global seismology,

hydraulic fracturing monitoring, shale oil/gas exploration, CO_2 injection monitoring, and many other fields of study. The main objective is to find the passive seismic source information, including source spatial location, onset, and seismic moment tensor, when giving a trustful velocity model. A potential contribution of OpenFWI to passive seismic problems is using the provided large number of simulated traces to train a neural network, such as PhaseNet [28], that can do arrival picking or event detection, which is an important step for passive seismic imaging problems. We take P-wave arrival picking as an example in this paper. We convert the Style-B dataset by adding (a) the labels of P-wave arrivals, and (b) strong Gaussian noise ($\sigma = 5 \times 10^{-3}$) to the seismic traces. Then we train an InversionNet to predict the P-wave arrivals from noisy seismic traces. The corresponding test results are shown in Figure 22, which indicates the InversionNet trained on the converted dataset accurately recognizes P-wave arrivals. Besides, people may also manually set the sources in the OpenFWI dataset as unknowns, though they are all located on the top surface, and invert the sources and velocity models in FWI schemes [29, 30] or by neural networks. In a future OpenFWI version, we may open-source passive seismic datasets and benchmarks upon data availability and approval of DOE and LANL.

16 Discussion

16.1 Past Version

We remark that several datasets with the same name have been used in previous publications. Specifically, FlatVel-A and CurveVel-A first emerged in [9]; FlatFault-A and CurveFault-A were generated in [11]; Style-A and Style-B were proposed in [3]. OpenFWI unified all datasets in the "*Vel*", "*Fault*" and the "*Style*" families in terms of data size and forward modeling parameters, also training parameters for the benchmarking results. Therefore there is a slight difference from the previously reported experimental results. From OpenFWI, we will maintain the datasets as presented now, and the future comparison should also be conducted with OpenFWI benchmarks.

16.2 Limitations

OpenFWI datasets: All datasets are synthesized from only a few prior knowledge (i.e., mathematical representations, natural images, or geological reservoir), therefore would inherently limit the representativeness and variability of the generated velocity maps. We also note that the "*Style Family*" datasets are excellent candidates for the inversion of field data in general cases. However, there may be some specific subsurface structures that are not covered by OpenFWI. Additionally, it would be better if OpenFWI can be validated with some field data for further evaluation.

OpenFWI benchmarks: There are two main limitations with the current benchmarks. One, the 3D FWI has limited literature and our benchmark is almost solitary. Furthermore, our evaluation strategy tests randomly selected channels, therefore is not ubiquitous. The other concern is that data-driven FWI has been flourishing with new advancements, we may not be able to compare all other most recent methods due to the unavailability of the codes associated.

References

[1] A Brougois, Marielle Bourget, Patriek Lailly, Michel Poulet, Patrice Ricarte, and Roelof Versteeg. Marmousi, model and data. In *EAEG workshop-practical aspects of seismic data inversion*, pages cp–108. European Association of Geoscientists & Engineers, 1990.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[3] Shihang Feng, Youzuo Lin, and Brendt Wohlberg. Multiscale data-driven seismic full-waveform inversion with field data study. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–14, 2021.

[4] J. Wagoner. 3D Geologic Modeling of the Southern San Joaquin Basin for the Westcarb Kimberlina Demonstration Project- A Status Report. Technical Report LLNL-TR-412487, Lawrence Livermore National Laboraotry (LLNL), Livermore, CA (United States), April 2009.

[5] David Alumbaugh, Michael Commer, Dustin Crandall, Erika Gasperikova, Shihang Feng, William Harbert, Yaoguo Li, Youzuo Lin, Savini Manthila Samarasinghe, and Xianjin Yang. Development of a multi-scale synthetic data set for the testing of subsurface CO_2 storage monitoring strategies. In *American Geophysical Union (AGU)*, 2021.

[6] Xianjin Yang, Thomas A Buscheck, Kayyum Mansoor, Zan Wang, Kai Gao, Lianjie Huang, Delphine Appriou, and Susan A Carroll. Assessment of geophysical monitoring methods for detection of brine and CO_2 leakage in drinking water aquifers. *International Journal of Greenhouse Gas Control*, 90:102803, 2019.

[7] Peter Moczo, Johan OA Robertsson, and Leo Eisner. The finite-difference time-domain method for modeling of seismic wave propagation. *Advances in geophysics*, 48:421–516, 2007.

[8] Björn Engquist and Andrew Majda. Absorbing boundary conditions for numerical simulation of waves. *Proceedings of the National Academy of Sciences*, 74(5):1765–1766, 1977.

[9] Yue Wu and Youzuo Lin. InversionNet: An efficient and accurate data-driven full waveform inversion. *IEEE Transactions on Computational Imaging*, 6:419–433, 2019.

[10] Zhongping Zhang and Youzuo Lin. Data-driven seismic waveform inversion: A study on the robustness and generalization. *IEEE Transactions on Geoscience and Remote sensing*, 58(10):6900–6913, 2020.

[11] Peng Jin, Xitong Zhang, Yinpeng Chen, Sharon Huang, Zicheng Liu, and Youzuo Lin. Unsupervised learning of full-waveform inversion: Connecting CNN and partial differential equation in a loop. In *Proc. Tenth International Conference on Learning Representations (ICLR)*, 2022.

[12] Qili Zeng, Shihang Feng, Brendt Wohlberg, and Youzuo Lin. InversionNet3D: Efficient and scalable learning for 3-D full-waveform inversion. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–16, 2022.

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[14] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *stat*, 1050:26, 2017.

[15] Gerard T Schuster. *Seismic inversion*. Society of Exploration Geophysicists, 2017.

[16] Honghai Yu and Stefan Winkler. Image complexity and spatial information. In *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 12–17. IEEE, 2013.

[17] Leida Li, Hao Cai, Yabin Zhang, Weisi Lin, Alex C Kot, and Xingming Sun. Sparse representation-based image quality index with adaptive sub-dictionaries. *IEEE Transactions on Image Processing*, 25(8):3775–3786, 2016.

[18] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[19] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 23(2):358–367, 1988.

[20] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

[21] Carey Bunks, Fatimetou M Saleck, S Zaleski, and G Chavent. Multiscale seismic waveform inversion. *Geophysics*, 60(5):1457–1473, 1995.

[22] Romain Brossier, Stéphane Operto, and Jean Virieux. Seismic imaging of complex onshore structures by 2d elastic frequency-domain full-waveform inversion. *Geophysics*, 74(6):WCC105–WCC118, 2009.

[23] Vincent Prieux, Gilles Lambaré, Stéphane Operto, and Jean Virieux. Building starting models for full waveform inversion from wide-aperture data by stereotomography. *Geophysical Prospecting*, 61:109–137, 2013.

[24] Debanjan Datta and Mrinal K Sen. Estimating a starting model for full-waveform inversion using a global optimization method. *Geophysics*, 81(4):R211–R223, 2016.

[25] Daniela Teodor, Cesare Comina, Farbod Khosro Anjom, Romain Brossier, Laura Valentina Socco, and Jean Virieux. Challenges in shallow target reconstruction by 3d elastic full-waveform inversion—which initial model? *Geophysics*, 86(4):R433–R446, 2021.

[26] Satinder Chopra and Glen Larsen. Acquisition footprint—its detection and removal. *CSEG Recorder*, 25(8):16–20, 2000.

[27] Yunsong Huang and Gerard T Schuster. Full-waveform inversion with multisource frequency selection of marine streamer data. *Geophysical Prospecting*, 66(7):1243–1257, 2018.

[28] Weiqiang Zhu and Gregory C Beroza. PhaseNet: a deep-neural-network-based seismic arrival-time picking method. *Geophysical Journal International*, 216(1):261–273, 2019.

[29] Ke Wang, Jerry R Krebs, Dave Hinkley, and Anatoly Baumstein. Simultaneous full-waveform inversion for source wavelet and earth model. In *2009 SEG Annual Meeting*. OnePetro, 2009.

[30] Hanchen Wang and Tariq Alkhalifah. Microseismic imaging using a source function independent full waveform inversion method. *Geophysical Journal International*, 214(1):46–57, 2018.