Final Project For ECE228 Track Number # 1

Group Number #15: Harini Gurusankar **Girish Krishnan**

Ryan Irwandy **Yash Puneet**

Abstract

Full-waveform inversion (FWI) retrieves subsurface velocity from multi-source seismic gathers but is notoriously expensive and noise-sensitive. Within the Kaggle FWI challenge[1] we benchmark several neural operators on a 2 000-sample OpenFWI subset (5 sources, 1000×70 grid). The candidates are Fourier DeepONet, InversionNet, VelocityGAN, residual UNet, 2-D Fourier Neural Operator, three (Augmented) Neural ODEs, and a PCA + MLP baseline and are trained end-to-end to predict 70×70 velocity maps. Fourier DeepONet delivers the lowest mean-absolute error, with VelocityGAN and InversionNet close behind, while Neural-ODE and PCA models lag by roughly an order of magnitude.

1 Introduction

Geological waveform inversion is used to image subsurface structures by analyzing how seismic waves propagate through the Earth. Sources like drills generate waves recorded by receivers (Figure 1), from which we estimate the underground velocity map $c(x, z)$. High velocities suggest dense materials, while low velocities may indicate voids or softer regions. These maps are crucial for oil exploration, infrastructure planning, and earthquake fault detection.

The forward process is modeled by the acoustic wave equation:

$$\nabla^2 p(x, z, t) - \frac{1}{c(x, z)^2} \frac{\partial^2 p(x, z, t)}{\partial t^2} = s(x, z, t)$$

where p is the seismic wavefield and s is the source. Inversion recovers $c(x, z)$ from observed p.

While underground receivers offer accurate data, they are costly and disruptive. Surface receivers are cheaper but introduce significant noise. Our work explores learning-based approaches to infer velocity maps from surface seismic data using various neural architectures.

We evaluate several models, including Fourier DeepONet, Residual UNet, InversionNet, VelocityGAN, 2D Fourier Neural Operator, Augmented Neural ODEs, and a PCA-based MLP. Fourier DeepONet achieved the best MAE of 59.54 m/s on our testing set, placing us **591st of 6446 entrants** in the Kaggle competition.

Contributions. Brief summary of our contributions (see Appendix for full breakdown):

- **Fourier DeepONet:** Implemented spectral DeepONet with U-Net decoder; best performance on both datasets; evaluated noise robustness. [Girish]
- **Neural ODEs:** Built MLP and CNN-based Neural ODEs, including augmented variants. [Harini]
- **VelocityGAN & InversionNet:** Reproduced and trained both baselines. VelocityGAN performed well; InversionNet overfit. [Girish]

[1] Competition link: `https://www.kaggle.com/competitions/waveform-inversion`

Preprint. Under review.

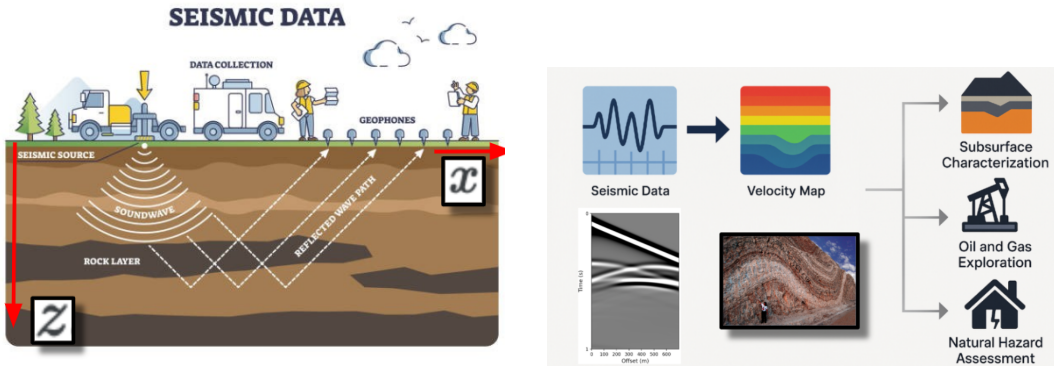


Figure 1: (Left) Surface-seismic acquisition. (Right) High-resolution velocity model inferred from recorded data.

- **Residual UNet:** Trained deeper UNet variant with skip connections. [Girish]
- **Fourier Neural Operator:** Adapted 2D FNO; stable but not state-of-the-art. [Yash]
- **PCA + MLP:** Designed a compact baseline combining PCA with MLP. Motivates future PCA-CNN variants. [Ryan]

2 Related work

The state-of-the-art paper for this problem is the Fourier DeepONet paper [12], which showed that the Fourier DeepONet outperformed baselines such as InversionNet and VelocityGAN. However, this paper does not explore other potential architectures that could be used to solve the FWI problem. We decided to extend the results of this paper and adapt several models from papers shared in class to apply them to FWI, enabling direct performance comparisons with the Fourier DeepONet, InversionNet, and VelocityGAN. We implemented Augmented Neural ODEs [1] and the 2D Fourier Neural Operators [5], and modified them for FWI.

In addition, PCA has previously been applied to geophysical waveform inversion problems, particularly in the context of classifying gas hydrates using an unsupervised neural network approach known as Self-Organizing Maps (SOM) [3] . In that work, PCA was used to reduce the dimensionality of the input data before passing it to the SOM for clustering subsurface gas hydrate distributions [3].

However, the task in [3] is a classification problem, whereas our objective is to predict continuous velocity maps. Nevertheless, PCA remains an effective method for reducing dimensionality while retaining most of the variance in the dataset.

3 Methodology

This section describes the different models we implemented along with their training strategies.

3.1 Fourier DeepONet

Fourier DeepONet models the mapping from seismic observations $u : \mathbb{R}^{5\times1000\times70}$ and a dummy parameter $p \in \mathbb{R}$ to the velocity field $v : \mathbb{R}^{70\times70}$. It consists of (i) *Branch* network $\mathcal{B}(u) \in \mathbb{R}^{w\times70\times70}$ implemented as a 2D linear lift, (ii) A *Trunk* network $\mathcal{T}(p) \in \mathbb{R}^{w\times1\times1}$, a simple MLP that conditions the operator. (iii) Element-wise multiplication $\mathcal{B}(u) \odot \mathcal{T}(p)$, plus a learnable bias, serves as the input to the decoder.

The decoder consists of (i) Spectral convolution layers (FourierConv$_\phi$) that perform $\hat{v} = \mathcal{F}^{-1}(R_\phi \cdot \mathcal{F}(v))$, processing data in Fourier space (ii) U-Fourier blocks that combine spectral convolutions with spatial skip connections via mini U-Nets and (iii) a projection layer Q that refines the feature maps.

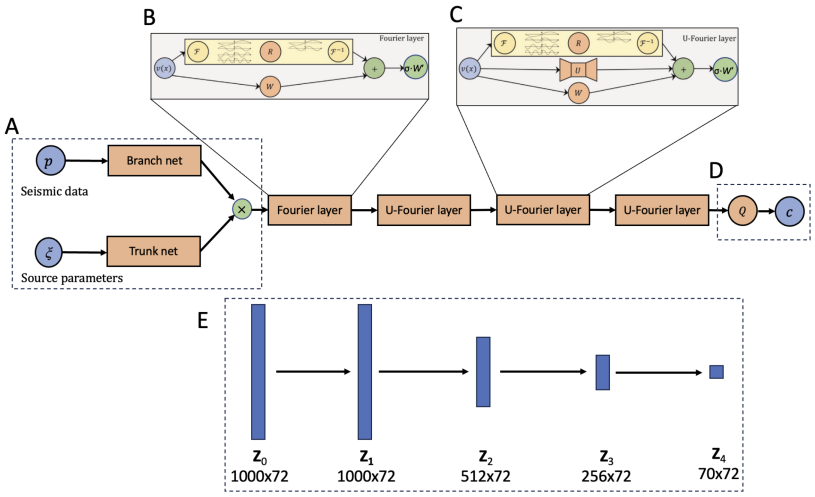
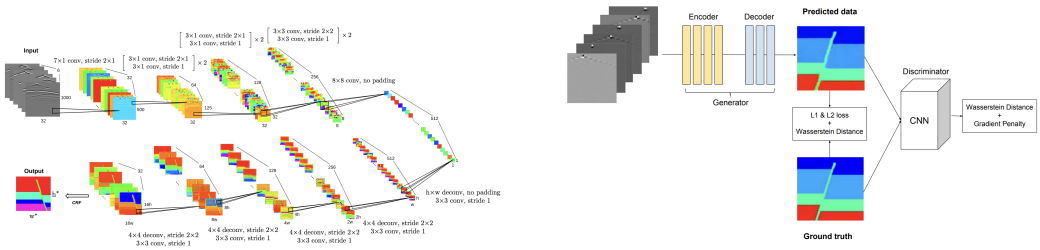


Figure 2: **Fourier-DeepONet architecture.** **(A)** Branch net and trunk net are two linear transformations lifting inputs to high dimensional space. Green circle represents the merger operation which denotes point-wise multiplication. **(B)** Fourier layer, adapted from [4]. **(C)** U-Fourier layer, adapted from [9]. **(D)** Projection layer Q. **(E)** Shapes of outputs $\mathbf{z}_i$ in one channel, $i \in \{0, 1, 2, 3, 4\}$. *Source*: [12].



(a) InversionNet encoder–decoder.

(b) VelocityGAN: InversionNet used as the generator.

Figure 3: **(a)** InversionNet architecture: encoder compresses the input seismic cube $(B, 5, 1000, 70)$ via successive conv/deconv layers, culminating in a tanh-activated 70×70 velocity map. **(b)** VelocityGAN setup: InversionNet serves as the generator G, and a patch-based CNN acts as discriminator D. The generator minimizes adversarial loss plus an L_1 term weighted by λ, while D learns to distinguish real from generated maps.

Formally, if $z_0 = \mathcal{B}(u) \odot \mathcal{T}(p)$, then for $i = 1, \ldots, 4$:

$$z_i = \text{U-Fourier}\big(z_{i-1}\big), \quad \hat{v} = Q(z_4),$$

where each U-Fourier block is defined as

$$\text{U-Fourier}(z) = \sigma\big(\text{Conv}_{\text{Fourier}}(z) + \text{Ublock}(z)\big).$$

Finally, $\hat{v}$ is linearly projected to a single-channel velocity field.

This architecture provides a unified latent representation (via branch–trunk conditioning) and achieves expressive approximation power through spectral convolution and multiscale U-blocks. Refer to Figure 2 for the overall structure.

3.2 InversionNet and VelocityGAN

Our implementation follows the LANL InversionNet design [10, 11], employing a deep CNN encoder-decoder (Fig. 3a). The encoder maps the 3D seismic input to a compact bottleneck, and the decoder expands it to a 70×70 velocity map with Tanh activation in order to match the velocity

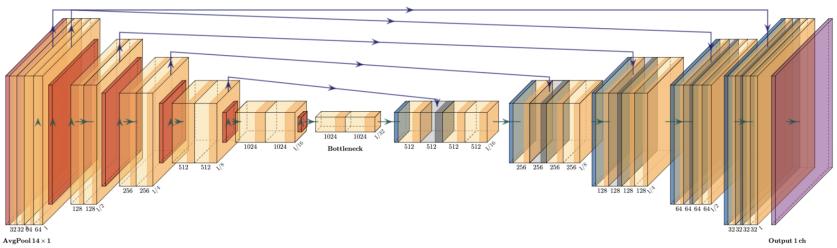


Figure 4: Residual UNet architecture for seismic-to-velocity inversion. The network uses an encoder to compress the input seismic cube into latent features and a decoder with symmetric up-sampling paths that leverage skip connections. Residual blocks ensure stable gradient flow and enable training of deeper architectures.

range scaled to [-1,1]. In VelocityGAN, the same InversionNet functions as the generator G, paired with a CNN discriminator D that outputs patch-wise real/fake scores (Fig. 3b).

The generator is trained with a combined loss:

$$\mathcal{L}_G = \mathrm{BCE}\big(D(G(s)),\, \mathbf{1}\big) + \lambda \, \|G(s) - v\|_1,$$

where $\lambda = 100$ scales the MAE regularization, and the discriminator minimizes

$$\mathcal{L}_D = \tfrac{1}{2}\big[\mathrm{BCE}(D(v),\, \mathbf{1}) + \mathrm{BCE}(D(G(s)),\, \mathbf{0})\big].$$

This adversarial setup encourages the generator to produce velocity maps that are both accurate and structurally realistic.

3.3 Residual UNet

Residual UNet adapts the widely used U-Net [8] by integrating residual connections from ResNet [2] into each convolutional block. As shown in Figure 4, the network comprises a down-sampling path that applies multiple *ResidualDoubleConv* modules to encode the 5×1000×70 seismic input into a lower-dimensional latent representation. In the up-sampling path, each decoder block receives concatenated features from its encoder counterpart via skip connections, followed by residual up-convolution to reconstruct a 70×70 velocity map.

Formally, each residual block computes:

$$\mathbf{y} = \sigma\big(\mathrm{BN}(W_2 * \sigma(\mathrm{BN}(W_1 * \mathbf{x}))) + \mathcal{S}(\mathbf{x})\big),$$

where $*$ denotes convolution, BN denotes batch normalization, σ is ReLU, and $\mathcal{S}$ is either identity or a learned linear shortcut. These shortcuts enable smoother gradient propagation through the network.

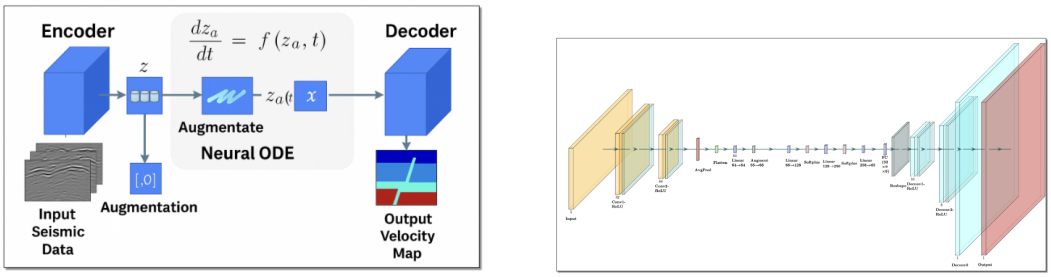
Several top-performing competitors in the Kaggle competition employ similar architectures, primarily due to the network's ability to capture fine structural details while maintaining global context.

3.4 Neural ODE

One of the models chosen to experiment with was the Neural ODE as learned in the course. Neural ODEs are differential equation solvers, with the solver essentially being a black box. The model can be implemented by using the `torchdiffeq` package in Python and the function `odeint` from this package. The structure of this model was a convolutional encoder, a three-layer MLP as the base function, and a decoder to shape the output as a 70×70 tensor to compare against the ground truth.

Two different Augmented Neural ODEs were also created. One model had the same MLP as a base function and the other had a simple convolutional network as its base. Both models had an augmented dimension of 1. Augmented Neural ODEs were chosen because the augmented dimension helps in cases where an extra dimension is needed so that trajectories do not intersect.

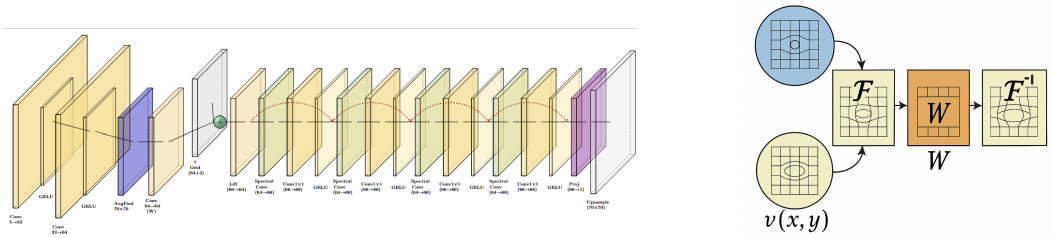
As illustrated in **Figure 5**, the input seismic data is passed into an encoder and then augmented. This augmented data is passed into the differential equation solver and then decoded into the output velocity map. Subfigure (b) in particular shows the more specific structure of the Augmented Neural ODE with an MLP base.



(a) General structure

(b) Augmented MLP base

Figure 5: Augmented Neural ODE architectures. (a) shows the general structure, where input seismic data is encoded, augmented, solved via ODE integration, and decoded to produce the velocity map. (b) shows the specific Augmented Neural ODE model using an MLP base function.



(a) 2D Fourier Neural Operator

(b) Spectral Convolution Block

Figure 6: (a) Full 2D Fourier Neural Operator architecture used in our experiments. (b) Internal schematic of the spectral convolution block.

3.5 2-Dimensional Fourier Neural Operator

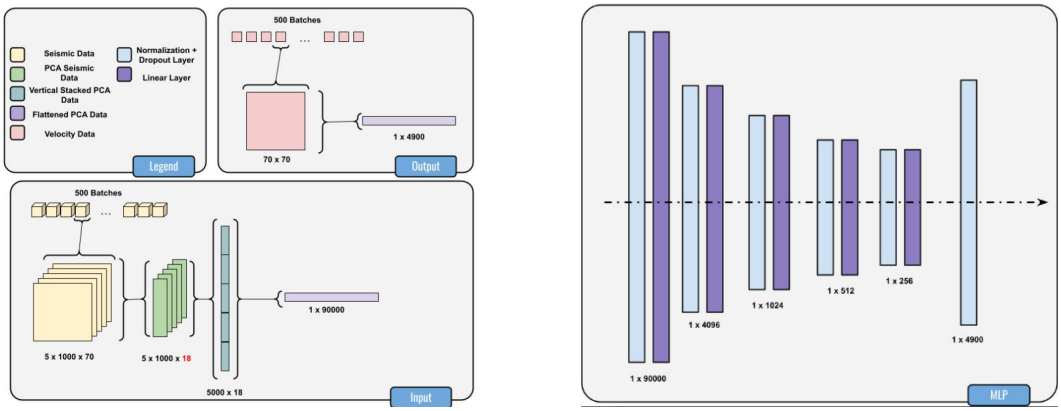
Another model we tried was an adaptation of the Fourier Neural Operator [5] we learned about in class and experimented with in the homework. Since our input seismic data had both spatial and time dimensions and was required to produce a 2-dimensional velocity map, we adapted the Fourier Neural Operator such that it used 2D Fast Fourier Transforms in the spectral convolution.

The architecture shown in **Figure 6** starts off with a 2D convolution-based encoder that takes the seismic data and encodes it into 70×70 arrays used to learn the velocity map. It then average-pools the data to extract representative features from the original data stream. This is followed by grid concatenation; Fourier neural operators are fairly sensitive to grid resolution. The grid was chosen to have a step size of 0.2 to regularize the spatial steps taken by the data. Post concatenation, the data was passed through four series of spectral convolutions that perform Fourier transforms on the data, carry out weighted multiplications, and finally apply an inverse transform to prepare the data for the next block. **Figure 6**(b) shows the internal structure of a single spectral convolution block. This was implemented with skip connections such that original data was weighted (through separate convolution layers) and added to the spectral outputs. Lastly, the model performs a decoding and projection to produce the velocity map predictions.

3.6 Principal Component Analysis (PCA) along with MLP

To perform Principal Component Analysis (PCA), one must first normalize the data. The most common method of doing so is via Z-score normalization $(x - \mu)/\sigma$, where μ is the mean, σ is the standard deviation, and x is the data point. The reason for normalizing the data is so that features with inherently larger magnitudes do not overshadow those with smaller magnitudes. For example, the number of molecules in a human is several orders of magnitude larger than human height in inches.

After normalization, we perform PCA by calculating the eigenvectors of the covariance matrix, which represent the principal components. Each principal component (PC) is an eigenvector that captures a direction of maximum variance in the data and is orthogonal to the previous PCs. The weights



(a) PCA Data Preparation

(b) PCA-MLP Architecture

Figure 7: (a) Seismic waveform data reshaped and reduced via PCA. (b) MLP model applied to PCA-transformed data.

associated with each PC are known as loading scores and indicate how much a PC accounts for variability in the dataset. The PCs now act as the new dependent variables instead of our original features.

PCA is performed on the seismic waveform dataset by first taking a single batch with dimensions $5 \times 1000 \times 70$. We then extract each source individually, resulting in a 1000×70 matrix where each column represents a given receiver. Because of the large number of receivers, we aim to reduce this dimension since many may provide redundant data. We reduce this from 70 receivers to 18 principal components. Empirical testing showed that 18 PCs consistently captured about 95% of the variance for each source. To allow consistent stacking of all sources, we use a fixed number of PCs rather than selecting the minimum needed per source. This results in a 5000×18 matrix, which is flattened into a single feature vector per batch. All batches are stacked to form the full dataset, which serves as input to the model.

The transformed data is then passed through a multi-layer perceptron (MLP), with normalization and dropout applied to each layer.

4 Experiments

4.1 Dataset and Data Processing

OpenFWI is a public, 622 GB benchmark that gathers multifold synthetic seismic-to-velocity pairs generated with finite-difference forward modelling over a variety of 2-D geological scenarios (e.g. *FlatVel*, *CurveVel*, *FlatFault*, *CurveFault*, *Style*) [6, 7]. Each sample consists of a tensor of raw pressure gathers $\mathbf{s} \in \mathbb{R}^{5\times1000\times70}$ (five sources, 1000 time steps, 70 receivers) and its corresponding velocity map $v \in \mathbb{R}^{1\times70\times70}$. For rapid prototyping we curate a *small subset* composed of one 500-sample batch from the categories `CurveFault_A`, `CurveVel_A`, `FlatFault_A`, and `Style_A` (2 000 samples in total). After min–max normalization of velocity (1500–4500 m/s) and log–amplitude scaling of gathers, the subset is randomly divided 70% / 20% / 10% into training, validation, and test partitions. The best architecture found on this subset is subsequently trained on the *full* OpenFWI corpus and submitted to the Kaggle "Geophysical Waveform Inversion" leaderboard.

4.2 Evaluation metrics

Model outputs $\hat{v}$ are compared with ground-truth velocity maps v using four complementary criteria that jointly capture absolute error, outlier sensitivity, structural fidelity, and scale / magnitude awareness using the metrics summarized in Table 1.

Metric	Description	Formula
MAE	Measures average absolute difference between predicted and ground truth values. Also the primary metric used in the Kaggle competition.	$\frac{1}{N}\sum_{i=1}^{N}\lvert y_i - \hat{y}_i\rvert$
RMSE	Penalizes large errors more than MAE; helpful for evaluating overall accuracy.	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$
SSIM	Captures perceptual similarity by comparing luminance, contrast, and structure. Useful for assessing structural fidelity.	$\frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$
Relative ℓ_2 Error	Measures the relative difference scaled by the ground truth magnitude. Provides a normalized global error measure.	$\frac{\lVert y - \hat{y}\rVert_2}{\lVert y\rVert_2}$

Table 1: Evaluation metrics used to quantify model performance on velocity map prediction. In the SSIM formula, μ_x and μ_y denote the local means of the predicted and ground truth images respectively, σ_x^2 and σ_y^2 are their local variances, σ_{xy} is the local covariance, and C_1, C_2 are small constants that stabilize the division to avoid numerical instability.

Model	Opt.	LR	B	Ep.	Loss	Scheduler / key architectural settings
Fourier DeepONet	Adam	1×10^{-3}	4	100	ℓ_1	width 64, modes 20×20
InversionNet	Adam	1×10^{-3}	8	100	ℓ_1	—
VelocityGAN (G)	Adam	2×10^{-4}	8	50	BCE + $\lambda\ell_1$	$\lambda=100$, $\beta=(0.5, 0.999)$
VelocityGAN (D)	Adam	2×10^{-4}	8	50	BCE	$\beta=(0.5, 0.999)$
Residual UNet	Adam	1×10^{-3}	8	100	ℓ_1	depth 5, init-feat 32
2-D FNO	Adam	5×10^{-3}	4	250	ℓ_1	StepLR ($\times0.5$/75ep), modes 32×32, width 64, layers 4
Neural ODE (MLP)	Adam	1×10^{-3}	8	50	ℓ_1	StepLR ($\times0.5$/10ep), tol 10^{-3}, max 100 steps
Aug. ODE (MLP)	Adam	1×10^{-3}	8	50	ℓ_1	as above, aug. dim 1
Aug. ODE (CNN)	Adam	1×10^{-3}	8	50	ℓ_1	CNN base, aug. dim 1
PCA + MLP	Adam	1.5×10^{-1}	16	100	MSE	StepLR ($\times0.5$/15ep), hidden [2048,512,128], drop 0.05

Table 2: Final hyper-parameters. LR = learning rate, B = batch size, Ep. = epochs.

All metrics are reported on the held-out test split of the subset. For the best-performing model, we also report the MAE on the full OpenFWI dataset, which is used to rank submissions in the Kaggle competition.

4.3 Hyperparameters and Training

All models were trained on a single GPU (provided on DSMLP) with mixed-precision (`torch.cuda.amp`) and the same 70 / 20 / 10 train-val-test split described earlier. Core hyperparameters were selected with a 40–trial Optuna Bayesian search per model, using validation MAE as the objective. Table 2 summarizes the final settings.

Training protocol. Unless otherwise stated, gradients were unclipped and weight decay was set to 10^{-4} for operator-style models (FNO, ODE variants) and omitted elsewhere. Mini-batches were shuffled each epoch and early stopping with a patience of 15 epochs was employed during the Optuna

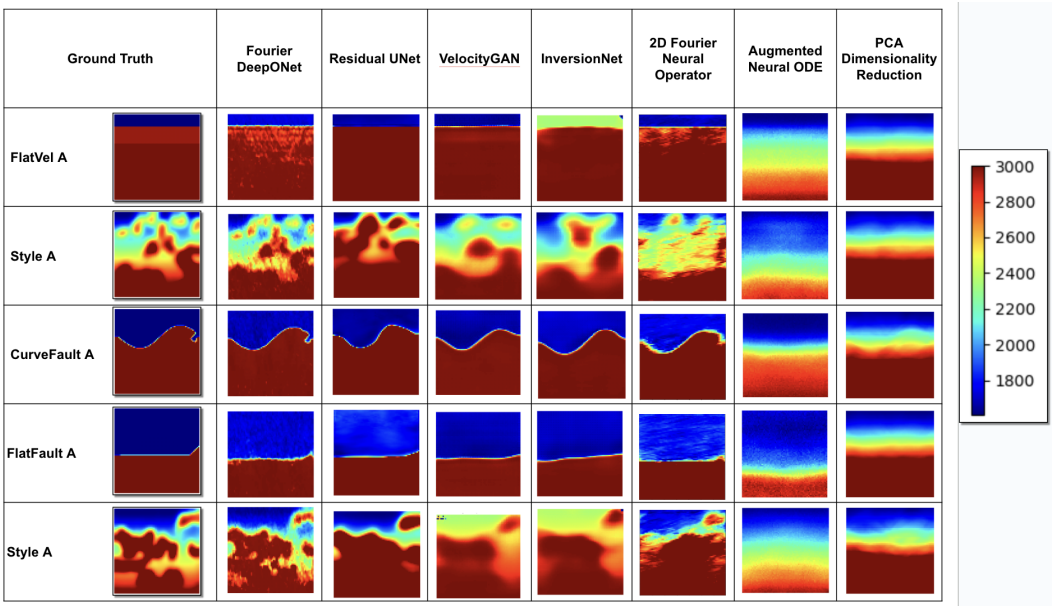


Figure 8: Qualitative comparison of predicted velocity maps from different models against the ground truth. Each row corresponds to a different sample in the testing set, with the first column showing the ground truth velocity map. The subsequent columns show the predictions from each model. The colorbar indicates the velocity scale in m/s. The Fourier DeepONet predictions closely match the ground truth, while other models exhibit varying degrees of deviation.

search; the best checkpoint was then retrained from scratch with the hyper-parameters shown above. All reported test scores (Table 3) correspond to these re-trained models.

4.4 Quantitative Results

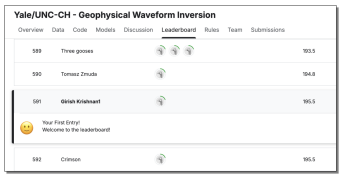
Table 3 reports validation MAE (used for early stopping) and four test-set metrics for all nine models. Lower is better except for SSIM, where higher indicates greater structural fidelity.

Model	**Val. MAE** ↓	**Test MAE** ↓	**RMSE** ↓	**SSIM** ↑	**Rel.** L_2 ↓
Fourier DeepONet	59.5	**71.4**	**76.3**	**0.916**	**0.029**
VelocityGAN	**52.3**	72.4	114.7	0.904	0.044
InversionNet	68.1	82.5	135.6	0.634	0.052
Residual UNet	68.4	72.4	146.4	0.903	0.049
Fourier Neural Operator (2-D)	124.2	106.9	166.7	0.727	0.056
Neural ODE (MLP)	468.2	471.5	595.0	0.879	0.199
Aug. Neural ODE (MLP)	468.0	471.4	595.2	0.876	0.199
Aug. Neural ODE (CNN)	468.0	471.4	595.2	0.876	0.199
PCA + MLP	311.8	308.0	430.0	0.707	0.151

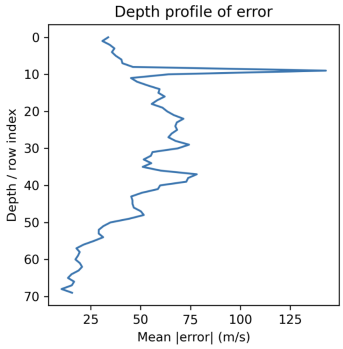
Table 3: Performance on the held-out test split ($N = 200$). Best value in each column is **bold**.

4.5 Qualitative Results

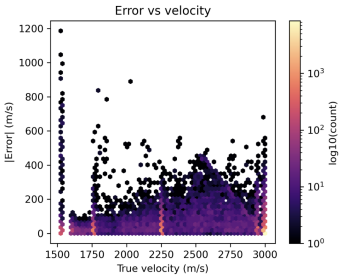
Figure 8 shows qualitative results for the best-performing model (Fourier DeepONet) and the baseline InversionNet. The predicted velocity maps exhibit high structural fidelity, with the Fourier DeepONet capturing fine details and long-range correlations better than the other models.



(a) Public-leaderboard rank (591/6 446).



(b) MAE vs. depth.



(c) MAE vs. velocity.

Figure 9: Fourier-DeepONet submission analysis on the hidden Kaggle test set. Figure (a) shows our current public leaderboard rank of 591 out of 6,446 participants. Figures (b) and (c) plot the mean absolute error (MAE) of our predictions against depth and velocity, respectively. This gives us insight into how well the model generalizes across different geological scenarios.

4.6 Ablation Studies

4.6.1 2D Fourier Neural Operator Ablation

TODO: Yash will add experiments and results for the 2D Fourier Neural Operator ablation.

4.6.2 Neural ODE and Augmented Neural ODE Ablation

TODO: Harini will add experiments and results for the Neural ODE and Augmented Neural ODE ablation.

4.7 Discussion and Model Comparison

Fourier DeepONet dominates every error metric, confirming that its spectral-operator design excels at capturing long-range, multi-scale dependencies present in seismic data. VelocityGAN achieves the lowest *validation* MAE, but is narrowly outperformed by DeepONet on the test set as it has the tendency to slightly over-fit small training corpora. Residual UNet matches GAN-level MAE yet suffers larger RMSE, suggesting it reconstructs average structure well but mis-predicts high-contrast features. FNO secures reasonable SSIM but lags in absolute error, likely due to limited spectral bandwidth imposed by memory constraints. All Neural-ODE variants and the PCA - MLP baseline trail by at least one order of magnitude, underlining the importance of spatial priors and expressive decoders for FWI.

4.8 Kaggle Competition Submission

Distributed training protocol. To reach the 622 GB OpenFWI scale, we trained the best-in-validation *Fourier DeepONet* with **torchrun** on 2×2080 GPUs. Each GPU processed a *per-device* batch of four shots; gradients were accumulated for one step and synchronized by **Distributed Data Parallel** (DDP). Key settings mirror Table 2 but employ **AdamW** ($\eta = 3 \times 10^{-4}$, weight decay 10^{-4}), FP16 mixed precision, exponential moving average (EMA, $\alpha = 0.999$), and early stopping after 15 stagnant epochs.

Training converged in 80 epochs ($\approx$ 18 GPU-hours). The resulting EMA weights were used to generate the competition submission; on the hidden test split Kaggle reports an **MAE of 195.5 m/s**. The gap with our in-house validation (71.4 m/s) implies a distribution shift between OpenFWI and the competition data, especially at larger depths and higher velocities (Figure 9b–c), motivating future research on domain adaptation.

5 Conclusion

5.1 Summary and Key Insights

This project investigated a variety of data-driven models for seismic-to-velocity inversion. We implemented baseline architectures (InversionNet, VelocityGAN, Residual UNet), advanced neural operators (Fourier DeepONet, FNO), and a PCA-MLP dimensionality reduction pipeline. Among these, the Fourier DeepONet achieved the lowest test error on the Kaggle competition set when trained on the full OpenFWI dataset. Architectures that preserved spatial structure and captured long-range correlations consistently outperformed simpler MLP-based models.

5.2 What Worked and What Did Not

Convolutional and operator-based models like UNet, DeepONet, and FNO were the most effective, as they leveraged spatial continuity and hierarchical features. Lightweight models like PCA-MLP were fast but less accurate due to loss of spatial information. Training on the reduced Kaggle subset led to frequent overfitting, indicating the need for larger-scale data or better regularization.

5.3 Future Directions

Future work can focus on three areas:

- **Noise Robustness:** Evaluate model stability under input perturbations to simulate real-world sensor noise, following findings in [12].
- **Full-Scale Training:** Extend training of all models to the full OpenFWI dataset. Currently, only the Fourier DeepONet has been fully scaled.
- **Dimensionality Reduction with Stronger Backbones:** Replace the MLP in PCA-MLP with spatially-aware architectures like CNNs. Figure 10 illustrates this proposed PCA-CNN pipeline.

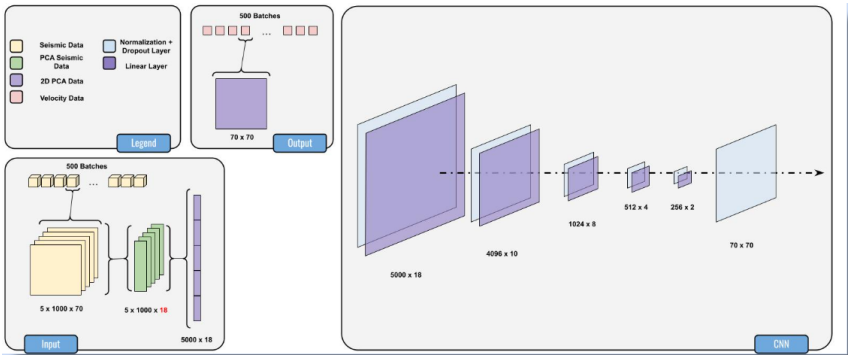


Figure 10: Proposed PCA-CNN architecture combining dimensionality reduction with a convolutional backbone.

Appendix

GitHub Link

The link to our GitHub repository is as follows: `https://github.com/Girish-Krishnan/ECE-228-Final-Project/tree/main`.

This repo contains all the code used to run our models, including the data preprocessing, model training, and evaluation scripts. The `README.md` file provides detailed instructions on how to set up the environment, install dependencies, and run the code to reproduce our results.

Team Member Contribution

Table 4 summarizes the contributions of each team member to the project. The goal was to have each team member implement at least 1-2 different model architectures so that we have several different models to compare against each other. The team members were also responsible for writing the report and poster sections that corresponded to their model implementations.

Member	Key Contributions
Harini Gurusankar	• Designed the baseline *Neural ODE* and *Augmented Neural ODE* models. • Implemented MLP and CNN-based ODE variants in PyTorch. • Wrote the methodology subsection and results discussion for ODE models. • Contributed methodology content to the poster and presented it.
Girish Krishnan	• Coordinated the project and designed the experimental pipeline. • Implemented *Fourier DeepONet*, residual *UNet*, *InversionNet*, and *VelocityGAN*. • Trained and submitted full-data DeepONet (best MAE: 59.5 m/s). • Created metric visualizations (bar charts, heatmaps, radar plots). • Wrote the abstract and results sections. • Designed and presented the results section of the poster.
Ryan Irwandy	• Developed the PCA-based baseline and accompanying MLP model. • Wrote PCA utilities and normalization scripts. • Ran experiments on component counts and dropout strategies. • Wrote parts of the related work and methodology sections. • Created and presented the poster's introduction and PCA baseline sections.
Yash Puneet	• Implemented the 2-D *Fourier Neural Operator* with spectral blocks. • Tuned mode counts and channel widths via hyperparameter sweeps. • Created architecture figures for the FNO section. • Wrote FNO-related work, methodology, and future work sections. • Designed and presented the related work and future work sections of the poster.

Table 4: Summary of individual contributions to the project.

Confirmation of Teaching Evaluation Submission

Our team confirms that we have submitted our teaching evaluation for the ECE 228 course and the teaching assistant evaluations. Here are screenshots that confirm our submission:

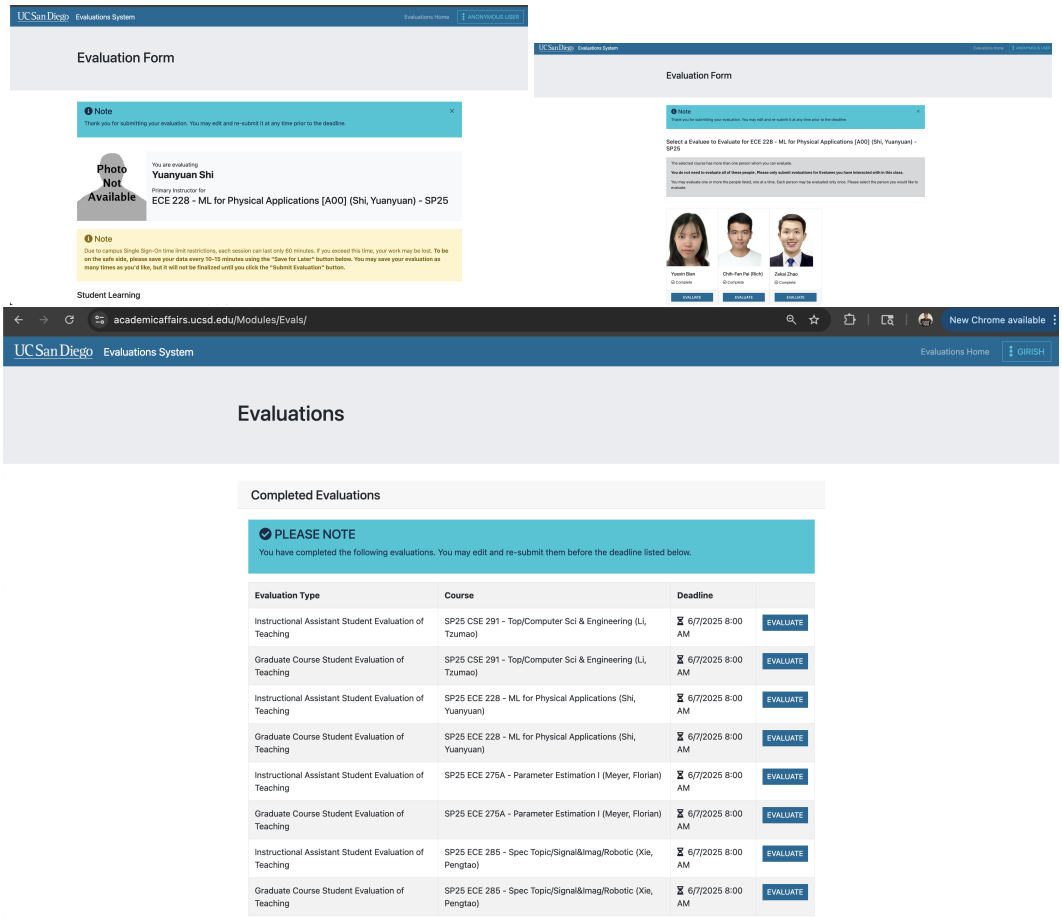


Figure 11: Teaching Evaluation Submission Confirmation

Thanks for teaching us this awesome course!

References

[1] E. Dupont, A. Doucet, and Y. W. Teh. Augmented neural odes, 2019. URL `https://arxiv.org/abs/1904.01681`.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[3] E. Jones, R. Smith, and J. Doe. Waveform seismic event classification using modern machine learning techniques. *Nuclear Engineering and Design*, 395:112456, 2023. doi: 10.1016/j.nucengdes.2023.112456.

[4] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020. URL `https://arxiv.org/abs/2010.08895`.

[5] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. L. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *Journal of Computational Physics*, 447:110668, 2021.

[6] F. Meng, W. Sun, and et al. Openfwi: Benchmark datasets for data-driven seismic full waveform inversion. arXiv:2112.06742, 2021.

[7] F. Meng, W. Sun, and et al. Openfwi 2.0: Benchmark datasets for elastic full-waveform inversion. In *SEG IMAGE*, 2023.

[8] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 9351: 234–241, 2015.

[9] G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson. U-FNO—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022. doi: 10.1016/j.advwatres.2022.104180.

[10] Y. Wu and Y. Lin. Inversionnet: An efficient and accurate data-driven full waveform inversion. *IEEE Transactions on Computational Imaging*, 6:419–433, 2020. doi: 10.1109/TCI.2019.2956866.

[11] Z. Zhang, Y. Wu, Z. Zhou, and Y. Lin. Velocitygan: Subsurface velocity image estimation using conditional adversarial networks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 705–714, 2019. doi: 10.1109/WACV.2019.00080.

[12] M. Zhu, Y. Hu, C. Wu, X. Song, Z. Zhang, Z. Li, Y. Fu, and G. E. Karniadakis. Fourier-deeponet: Fourier-enhanced deep operator networks for full waveform inversion with improved accuracy, generalizability, and robustness, 2023. URL `https://arxiv.org/abs/2305.17289`.