Non-peer reviewed **Research Article** submitted to SEISMICA

Supplementary Material for "Comprehensive Exploration of Machine Learning Approaches to Seismic Event Discrimination in the Pacific Northwest"

Akash Kharita * [1], **Marine Denolle** [1], **Alexander R. Hutko** [1,2], **J. Renate Hartog** [1,2], **Stephen D. Malone** [1,2]

[1]Earth and Space Sciences, University of Washington, Seattle, USA, [2]Pacific Northwest Seismic Network, University of Washington, Seattle, USA

S1 Scatnet Feature Extraction

The scattering coefficients are derived using a two-layer scattering network designed to capture hierarchical, multi-scale representations of the seismic waveforms. The network relies on Morlet wavelets as its basis functions, decomposing the input signal into its time-frequency components. The parameters of the scattering network are meticulously chosen to balance resolution and sensitivity to signal variations.

In the first layer, the octaves parameter determines the number of dyadic frequency bands analyzed. For instance, with five octaves, the frequency range is divided into progressively narrower bands, enabling the network to detect both low- and high-frequency components. The resolution parameter, set to two, dictates the granularity of the frequency band division within each octave, increasing the frequency resolution. The quality factor, set to one in the first layer, controls the trade-off between time and frequency localization, ensuring sharp temporal responses for precise event detection.

The second layer builds upon the first by considering interactions between frequency bands captured in the first layer. Here, the octaves, resolution, and quality are redefined (e.g., quality set to three), allowing the layer to focus on broader and smoother modulations of the signal. This layer computes coefficients that encode second-order interactions, representing how different frequency bands influence each other over time. These parameters are particularly crucial for capturing non-linearities and complex seismic event signatures, such as those arising from mixed-source phenomena or overlapping events.

The scattering coefficients at each layer are computed as the modulus of the wavelet transform, aggregated over time using operations like max pooling to summarize the signal's behavior over each time segment. The total number of wavelets (e.g., octaves × resolution) and their specific bandwidths ensure that the scattering coefficients offer a rich representation of the signal's energy distribution and dynamics across scales. These coefficients are normalized using logarithmic scaling, ensuring stability and robustness, particularly for signals with wide-ranging amplitudes. This parameterization allows the scattering network to effectively capture the complex, multi-scale features essential for characterizing seismic events in diverse geohazard contexts.

S2 Hyper-parameter tuning of CML models

In this study, we evaluated the performance of various machine learning algorithms on a dataset comprising 1000 traces per class, utilizing physical and TSFEL-generated features. For each algorithm, we conducted an exhaustive search over all possible hyperparameter combinations defined within a grid for each respective model. We employed five-fold cross-validation for every hyperparameter combination to ensure robustness, recording the resulting F1 scores to assess model performance.

MLP (Multi-Layer Perceptron) To tune its performance, the Multi-Layer Perceptron (MLP) was configured with several hyperparameters. The `hidden_layer_sizes` parameter specifies the number of neurons in each hidden layer, such as `(100)`, indicating one hidden layer with 100 neurons. Larger layer sizes may enhance the model's capacity to capture complex patterns, but can lead to overfitting if not tuned properly. The `activation` function was set to either `relu` (Rectified Linear Unit) or `tanh` (Hyperbolic Tangent). While `relu` offers efficient learning by setting negative inputs to zero, `tanh` maps inputs to a range between -1 and 1, which can help when centered outputs are needed. The `solver` for optimization was fixed to `adam`, a widely used method that combines the advantages of RMSProp and Stochastic Gradient Descent (SGD). Finally, the `max_iter` parameter was set to `500`, limiting the maximum number of iterations for convergence during training.

Grid for MLP

```
mlp_param_grid = {‘hidden_layer_sizes’: [(100,),
    (200,), (300,)],
 ‘activation’: [‘relu’, ‘tanh’],
 ‘solver’: [‘adam’],
 ‘max_iter’: [500]}
```

Optimal hyperparameter values for MLP were found to be as follows -

*Corresponding author: ak287@uw.edu

```
{‘activation’: ‘relu’, ‘hidden_layer_sizes’:
    (200,), ‘max_iter’: 500, ’solver’: ‘adam’}
```

SVC (Support Vector Classifier) The Support Vector Classifier (SVC) employed two key hyperparameters for tuning. The `C` parameter controls the regularization strength, with smaller values enforcing stronger regularization to reduce overfitting and larger values allowing for more flexible decision boundaries. The `kernel` parameter determines how the algorithm maps the input features into higher-dimensional spaces. Two kernel options were explored: `linear`, which fits a hyperplane to separate classes in the original feature space, and `rbf` (Radial Basis Function), which uses a Gaussian-based method for projecting data into higher dimensions, enabling the classifier to handle non-linear separable data.

Grid for SVC

```
svc_param_grid = {‘C’: [0.1, 1, 10], ‘kernel’:
    [‘linear’, ‘rbf’]}
```

Optimal hyperparameter values for SVC were found to be as follows -

```
{‘C’: 10, ‘kernel’: ‘rbf’}
```

KNN (K-Nearest Neighbors) The K-Nearest Neighbors (KNN) algorithm was tuned using the `n_neighbors` parameter, which specifies the number of neighbors to consider when making classification decisions. Smaller values result in more localized decision boundaries, potentially increasing sensitivity to noise, while larger values produce smoother decision surfaces.

Grid for KNN

```
knn_param_grid = {‘n_neighbors’: [3, 5, 7, 9]}
```

Optimal hyperparameter values for KNN were found to be as follows -

```
{‘n_neighbors’: 7}
```

Logistic Regression (LR) Logistic Regression was configured with three hyperparameters. The C parameter, representing the inverse of regularization strength, was explored over a range of values. Smaller values enforce stronger regularization to mitigate overfitting, whereas larger values allow the model to capture more intricate relationships. The `penalty` was fixed to `l2` (Ridge regularization), which adds a squared magnitude of coefficients as a penalty term. Two solvers were considered for optimization: `lbfgs`, a quasi-Newton method efficient for small- to medium-sized datasets, and `liblinear`, which works well for L2-regularized problems in smaller datasets. Grid for LR

```
lr_param_grid = {‘C’: [0.01, 0.1, 1, 10, 100],
    ‘penalty’: [‘l2’], ‘solver’: [‘lbfgs’,
    ‘liblinear’]}
```

Optimal hyperparameter values for LR were found to be as follows -

```
 {’C’: 0.01, ’penalty’: ’l2’, ’solver’: ’lbfgs’}
```

Random Forest (RF) The Random Forest model was tuned using two main parameters. The `n_estimators` parameter determines the number of decision trees in the forest. Increasing this number typically improves accuracy but also adds to the training time. The `max_depth` parameter defines the maximum depth of each tree. Deeper trees can better model complex relationships but may overfit the training data.

Grid for RF

```
rf_param_grid = {‘n_estimators’: [100, 200, 300],
    ‘max_depth’: [None, 10, 20]}
```

Optimal hyperparameter values for RF were found to be as follows -

```
{‘max_depth’: None, ‘n_estimators’: 300}
```

XGBoost (XGB) For the XGBoost algorithm, the hyperparameters `n_estimators` and `max_depth` were optimized. The `n_estimators` parameter specifies the number of boosting rounds or trees. Higher values generally improve performance but increase computational cost. The `max_depth` parameter limits the depth of individual trees, with deeper trees capable of capturing more intricate patterns at the expense of a higher risk of overfitting.

Grid for XGB

```
xgb_param_grid ={‘n_estimators’: [50, 100, 150],
    ‘max_depth’: [3, 5, 7]}
```

Optimal hyperparameter values for XGB were found to be as follows -

```
{‘max_depth’: 7, ‘n_estimators’: 100}
```

LightGBM (LGBM) The LightGBM model was configured with three hyperparameters. The `n_estimators` parameter, similar to XGBoost, determines the number of boosting rounds or trees. The `max_depth` parameter restricts the depth of the trees, with higher values enabling the model to learn more complex patterns. Finally, the `num_leaves` parameter sets the maximum number of leaves per tree, with a default of 31, balancing model complexity and computational efficiency.

By systematically exploring these hyperparameter grids, the study aimed to determine the optimal configuration for each model, maximizing F1 scores and enhancing classification performance on the multi-class dataset.

Grid for LGBM

```
lgbm_param_grid = {‘n_estimators’: [10, 50, 100],
    ‘max_depth’: [3, 5], ‘num_leaves’:[7]}
```

Optimal values LGBM were found to be as follows -

```
{‘max_depth’: 3, ‘n_estimators’: 50,
    ‘num_leaves’: 7}
```

Table S1 Physical Features Definitions

#	Feature	Formula
Waveform / envelope / autocorrelation:		
1	Window_Length	$t[-1] - t[0]$
2	RappMaxMean	$\mathrm{max(env)/mean(env)}$
3	RappMaxMedian	$\mathrm{max(env)/median(env)}$
4	AsDec	$(t[\arg\max(\mathrm{env})] - t[0])/(t[-1] - t[\arg\max(\mathrm{env})])$
5	KurtoSig	$\mathrm{kurtosis(data)}$
6	KurtoEnv	$\mathrm{kurtosis(env)}$
7	SkewSig	$\mathrm{skew(data)}$
8	SkewEnv	$\mathrm{skew(env)}$
9	CorPeakNumber	$\#\,\mathrm{peaks(auto)}$
10	Energy1/3Cor	$\int \mathrm{auto}[0:N/3]\; dt$
11	Energy2/3Cor	$\int \mathrm{auto}[N/3:N]\; dt$
12	int_ratio	$\dfrac{\int \mathrm{auto}[0:N/3]\; dt}{\int \mathrm{auto}[N/3:N]\; dt}$
13	RMSDecPhaseLine	$\sqrt{\mathrm{mean}((\mathrm{env} - \ell)^2)}$
Spectral (FFT-based):		
14	MeanFFT	$\mathrm{mean(ft)}$
15	MaxFFT	$\mathrm{max(ft)}$
16	FMaxFFT	$\mathrm{freq}[\arg\max(\mathrm{ft})]$
17	MedianFFT	$\mathrm{median(norm\_ft)}$
18	VarFFT	$\mathrm{var(norm\_ft)}$
19	FCentroid	$\dfrac{\sum \mathrm{freq} \cdot \mathrm{ft}}{\sum \mathrm{ft}}$
20	Fquart1	$\dfrac{\sum \mathrm{freq}_{0:N/4} \cdot \mathrm{ft}_{0:N/4}}{\sum \mathrm{ft}_{0:N/4}}$
21	Fquart3	$\dfrac{\sum \mathrm{freq}_{N/2:3N/4} \cdot \mathrm{ft}_{N/2:3N/4}}{\sum \mathrm{ft}_{N/2:3N/4}}$
22	NPeakFFT	$\#\,\mathrm{peaks(ft\ ;\ height} = 0.75\max(\mathrm{ft}))$
23	MeanPeaksFFT	$\mathrm{mean(ft[peaks(ft)])}$
24	E1FFT	$\int \mathrm{ft}_{0:N/4}\; d\mathrm{freq}$
25	E2FFT	$\int \mathrm{ft}_{N/4:N/2}\; d\mathrm{freq}$
26	E3FFT	$\int \mathrm{ft}_{N/2:3N/4}\; d\mathrm{freq}$
27	E4FFT	$\int \mathrm{ft}_{3N/4:N}\; d\mathrm{freq}$
28	Gamma1	$\dfrac{\sum \mathrm{freq} \cdot \mathrm{ft}^2}{\sum \mathrm{ft}^2}$
29	Gamma2	$\sqrt{\dfrac{\sum \mathrm{freq}^2 \cdot \mathrm{ft}^2}{\sum \mathrm{ft}^2}}$
30	Gamma	$\sqrt{\left(\dfrac{\sum \mathrm{freq} \cdot \mathrm{ft}^2}{\sum \mathrm{ft}^2}\right)^2 - \left(\sqrt{\dfrac{\sum \mathrm{freq}^2 \cdot \mathrm{ft}^2}{\sum \mathrm{ft}^2}}\right)^2}$
Time–frequency (DFT / spectrogram-based):		
31	KurtoMaxDFT	$\mathrm{kurtosis}(\max_\omega \lvert\mathrm{Sxx}\rvert)$
32	KurtoMedianDFT	$\mathrm{kurtosis}(\mathrm{median}_\omega \lvert\mathrm{Sxx}\rvert)$
33	MaxOverMeanDFT	$\mathrm{mean}\left(\dfrac{\max_\omega \lvert\mathrm{Sxx}\rvert}{\mathrm{mean}_\omega \lvert\mathrm{Sxx}\rvert}\right)$
34	MaxOverMedianDFT	$\mathrm{mean}\left(\dfrac{\max_\omega \lvert\mathrm{Sxx}\rvert}{\mathrm{median}_\omega \lvert\mathrm{Sxx}\rvert}\right)$
35	NbrPeaksMaxDFT	$\#\,\mathrm{peaks}(\max_\omega \lvert\mathrm{Sxx}\rvert)$
36	NbrPeaksMeanDFT	$\#\,\mathrm{peaks}(\mathrm{mean}_\omega \lvert\mathrm{Sxx}\rvert)$
37	NbrPeaksMedianDFT	$\#\,\mathrm{peaks}(\mathrm{median}_\omega \lvert\mathrm{Sxx}\rvert)$
38	45/46	$\dfrac{\#\,\mathrm{peaks}(\max_\omega \lvert\mathrm{Sxx}\rvert)}{\#\,\mathrm{peaks}(\mathrm{mean}_\omega \lvert\mathrm{Sxx}\rvert)}$
39	45/47	$\dfrac{\#\,\mathrm{peaks}(\max_\omega \lvert\mathrm{Sxx}\rvert)}{\#\,\mathrm{peaks}(\mathrm{median}_\omega \lvert\mathrm{Sxx}\rvert)}$
40	NbrPeaksCentralFreq	$\#\,\mathrm{peaks}\left(\dfrac{\sum f\, \lvert\mathrm{Sxx}\rvert}{\sum \lvert\mathrm{Sxx}\rvert}\right)$
41	NbrPeaksMaxFreq	$\#\,\mathrm{peaks}(f[\arg\max_\omega \lvert\mathrm{Sxx}\rvert])$
42	50/51	$\dfrac{\#\,\mathrm{peaks(central\_freq)}}{\#\,\mathrm{peaks(max\_freq)}}$
43	DistMaxMeanFreqDTF	$\mathrm{mean}(\max_\omega \lvert\mathrm{Sxx}\rvert - \mathrm{mean}_\omega \lvert\mathrm{Sxx}\rvert)$
44	DistMaxMedianFreqDTF	$\mathrm{mean}(\max_\omega \lvert\mathrm{Sxx}\rvert - \mathrm{median}_\omega \lvert\mathrm{Sxx}\rvert)$
45	DistQ2Q1DFT	$\mathrm{mean}\left(\dfrac{\sum f\, \mathrm{Sq}_2}{\sum \mathrm{Sq}_2} - \dfrac{\sum f\, \mathrm{Sq}_1}{\sum \mathrm{Sq}_1}\right)$
46	DistQ3Q2DFT	$\mathrm{mean}\left(\dfrac{\sum f\, \mathrm{Sq}_3}{\sum \mathrm{Sq}_3} - \dfrac{\sum f\, \mathrm{Sq}_2}{\sum \mathrm{Sq}_2}\right)$
47	DistQ3Q1DFT	$\mathrm{mean}\left(\dfrac{\sum f\, \mathrm{Sq}_3}{\sum \mathrm{Sq}_3} - \dfrac{\sum f\, \mathrm{Sq}_1}{\sum \mathrm{Sq}_1}\right)$
Envelope summary features:		
48	Peak_Envelope_Amplitude	$\mathrm{max(env)}$
49	Average_Envelope_Amplitude	$\mathrm{mean(env)}$
50	Envelope_Area	$\mathrm{AUC}(t, \mathrm{env})$
51	Envelope_Velocity	$\mathrm{AUC}(t, \mathrm{env})/(t[-1] - t[0])$
52	Envelope_Rise_Time	$t[\arg\max(\mathrm{env})] - t[0]$

Table S2 Highly correlated feature pairs removed to reduce redundancy. Here, r denotes the *absolute Pearson correlation coefficient* between the two features. For each pair with $r \geq 0.95$, we retained Feature 1 and removed Feature 2.

Feature 1 (kept)	Feature 2 (removed)	r
RMSDecPhaseLine	AsDec	0.960425
MeanPeaksFFT	MeanFFT	0.967360
Gamma1	FCentroid	0.955020
Gamma2	FCentroid	0.980566
Gamma2	Gamma1	0.985910
DistMaxMedianFreqDTF	DistMaxMeanFreqDTF	0.999963
Envelope_Area	Average_Envelope_Amplitude	1.000000
Envelope_Velocity	Average_Envelope_Amplitude	1.000000
Envelope_Velocity	Envelope_Area	1.000000

Note: The table lists time-domain, frequency-domain, and time–frequency attributes (physical features) computed from a windowed single component seismogram. Here, $s(t)$ denotes the windowed raw seismic trace, $e(t)$ its amplitude envelope, $a(\tau)$ the normalized autocorrelation of $s(t)$, and $S(\nu)$ the Fourier amplitude spectrum of $s(t)$ with frequency ν and peak frequency $\nu_{\max}$. For time–frequency features, $\mathrm{DFT}(t, \omega)$ (or Sxx) denotes the magnitude of the discrete short-time Fourier transform/spectrogram at time t and angular frequency ω; “max/mean/median” refer to reductions across frequency at each time. t_s and t_e are the window start and end times, and $t_{\max}$ is the time of peak envelope amplitude.

References

Ni, Y., Hutko, A., Skene, F., Denolle, M., Malone, S., Bodin, P., Hartog, R., and Wright, A. Curated Pacific Northwest AI-ready seismic dataset. 2023.

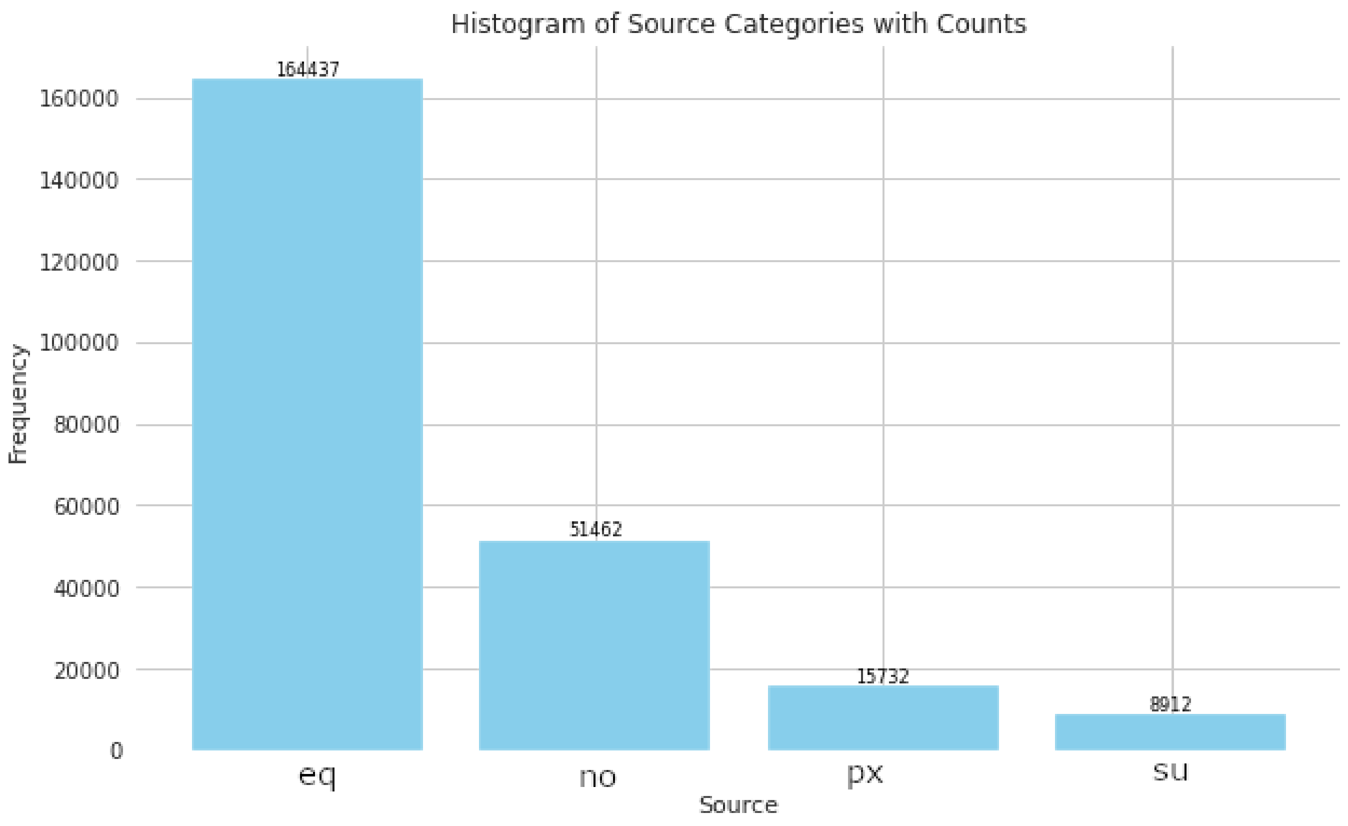

Figure S1 Number of waveforms available from Ni et al. (2023).

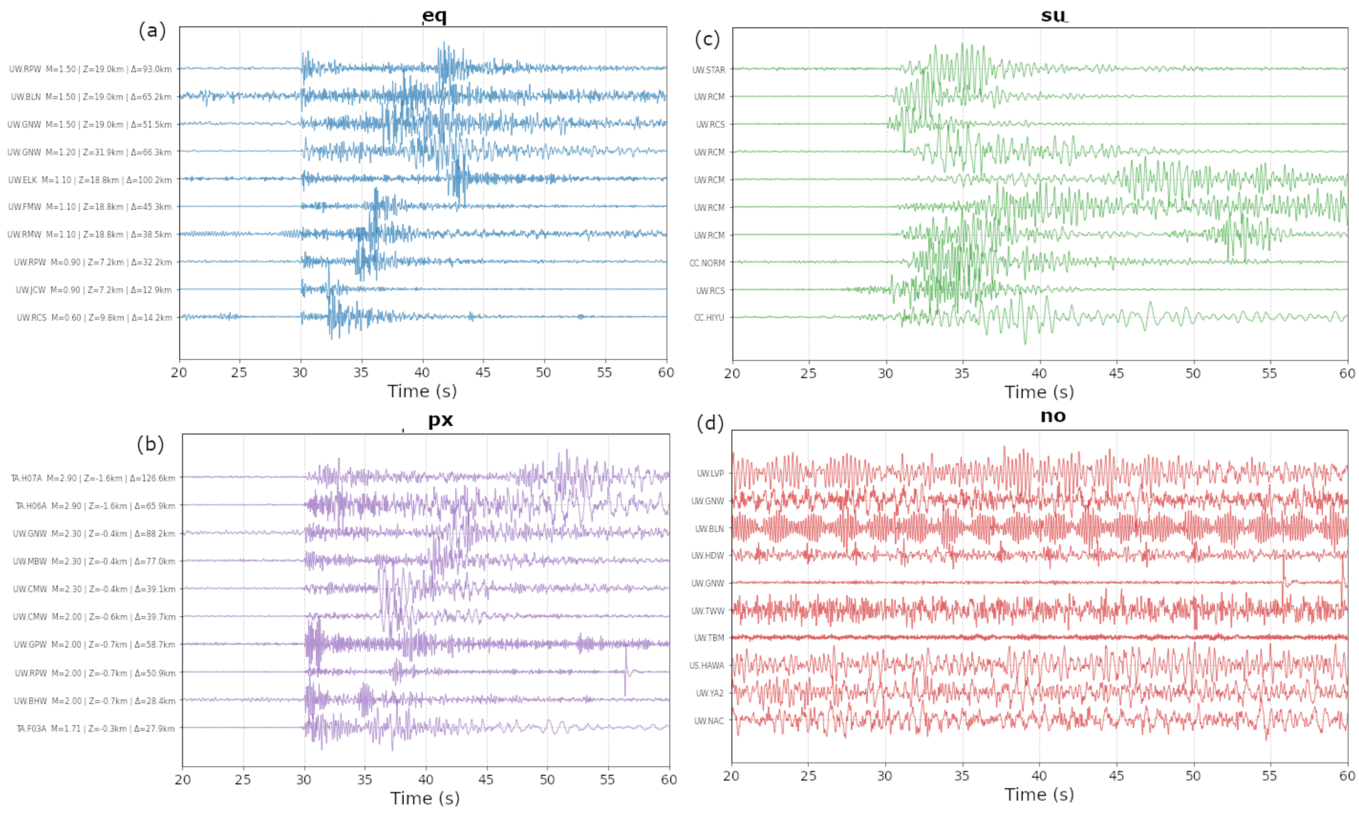

Figure S2 Representative waveforms from the curated test set. Example normalized three-component (vertical-channel) waveform windows for (a) earthquakes, (b) surface events, (c) explosions, and (d) noise. Each trace is time-aligned and vertically offset for clarity. For earthquakes and explosions, the left-hand labels report the catalog magnitude (M), source depth (Z, km), and epicentral distance (Δ, km) for the corresponding record; surface-event and noise examples are shown without catalog metadata.

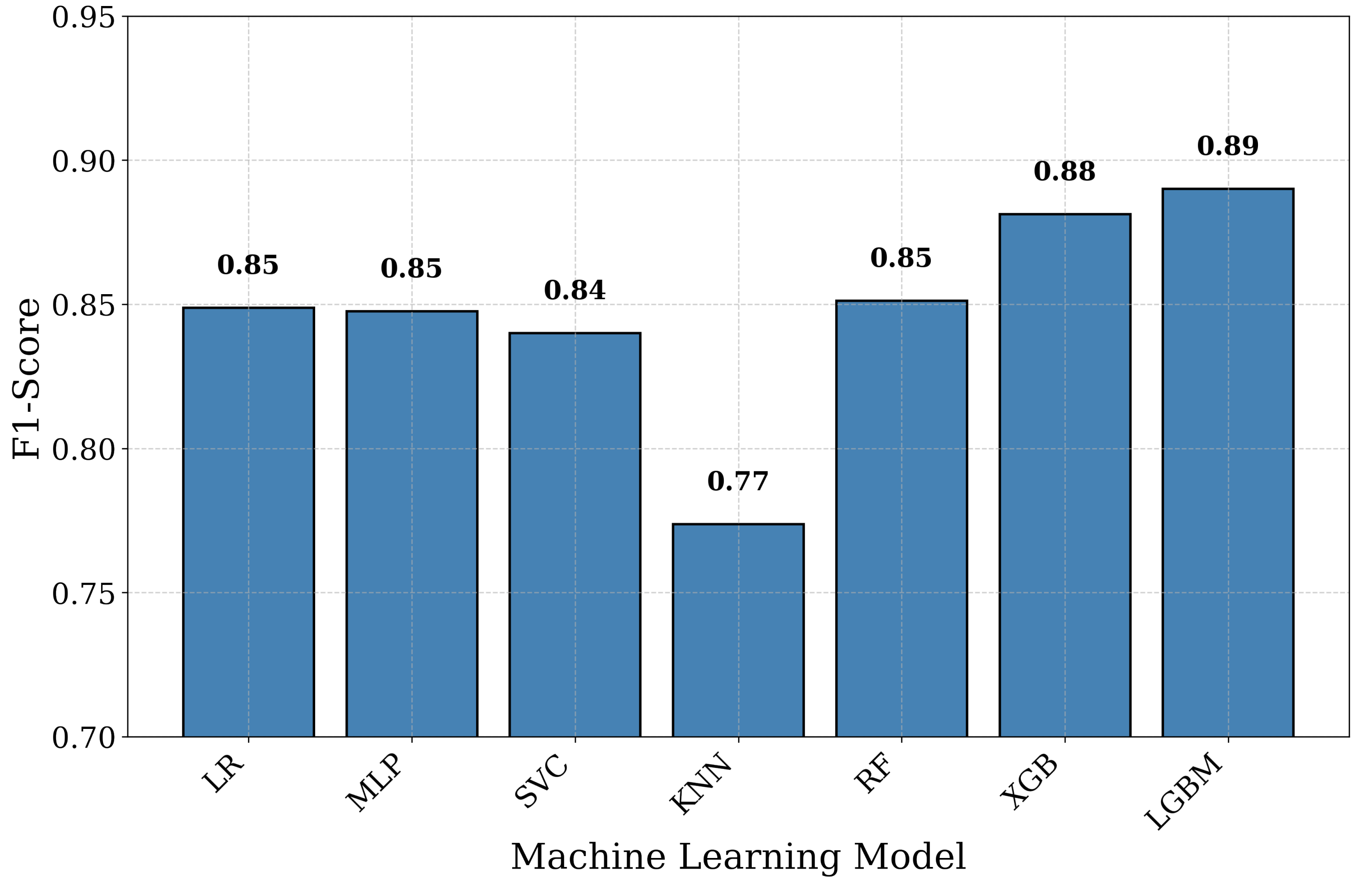

Figure S3 **F1 Score** of CML models tested and estimated on the test data from the curated data set.

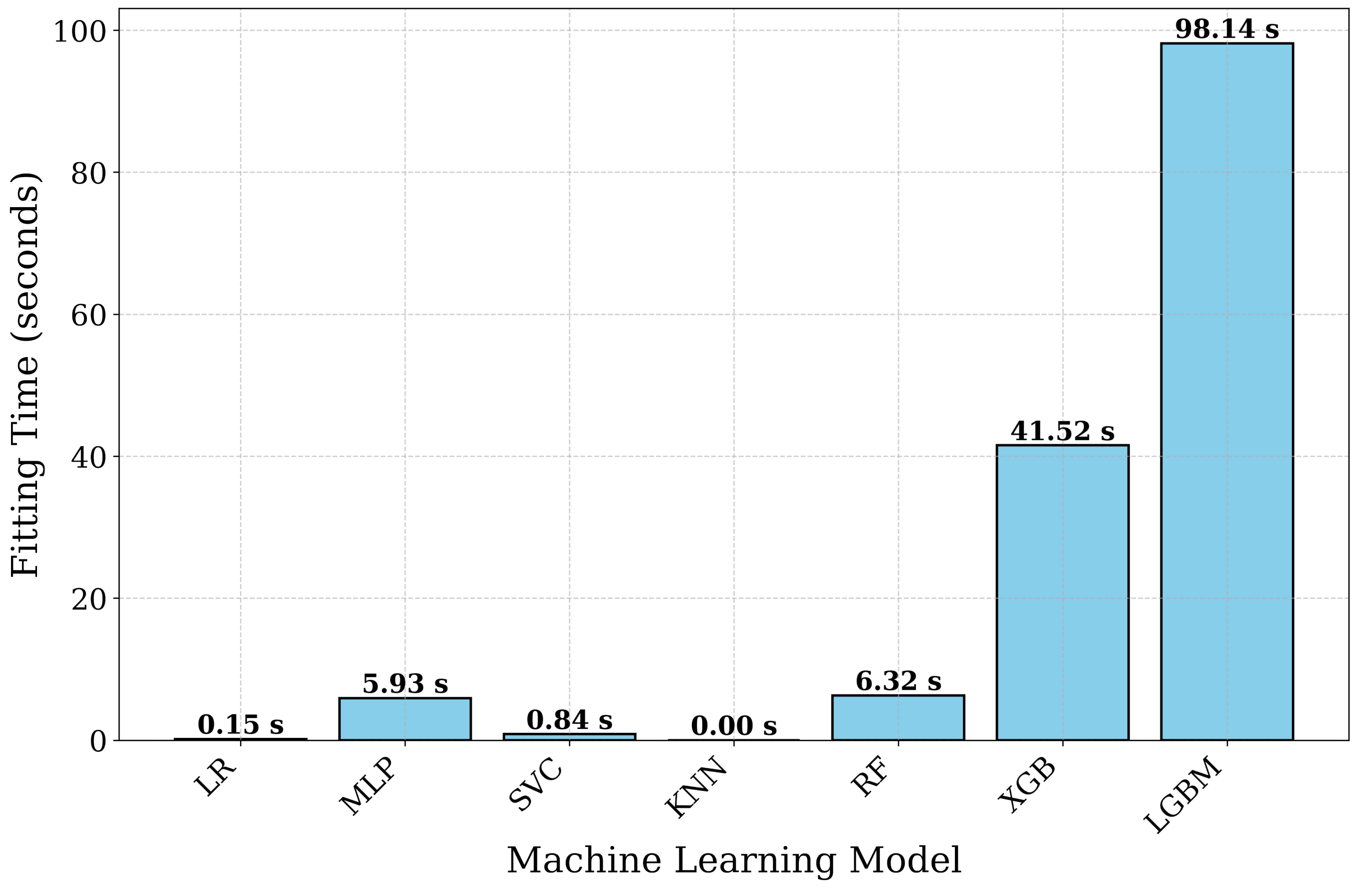

Figure S4 **Computational time for training the CML models.**

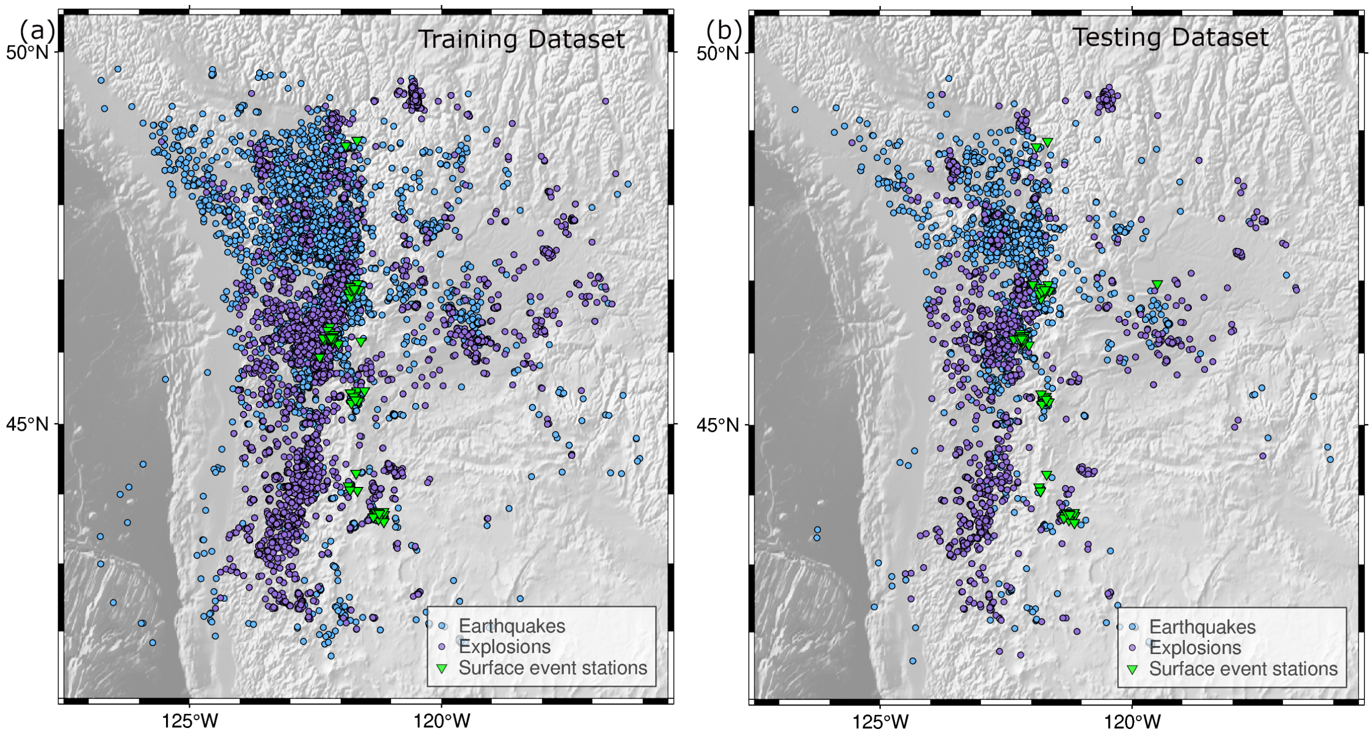

Figure S5 Spatial distribution of various classes in the training and testing dataset for Machine Learning models

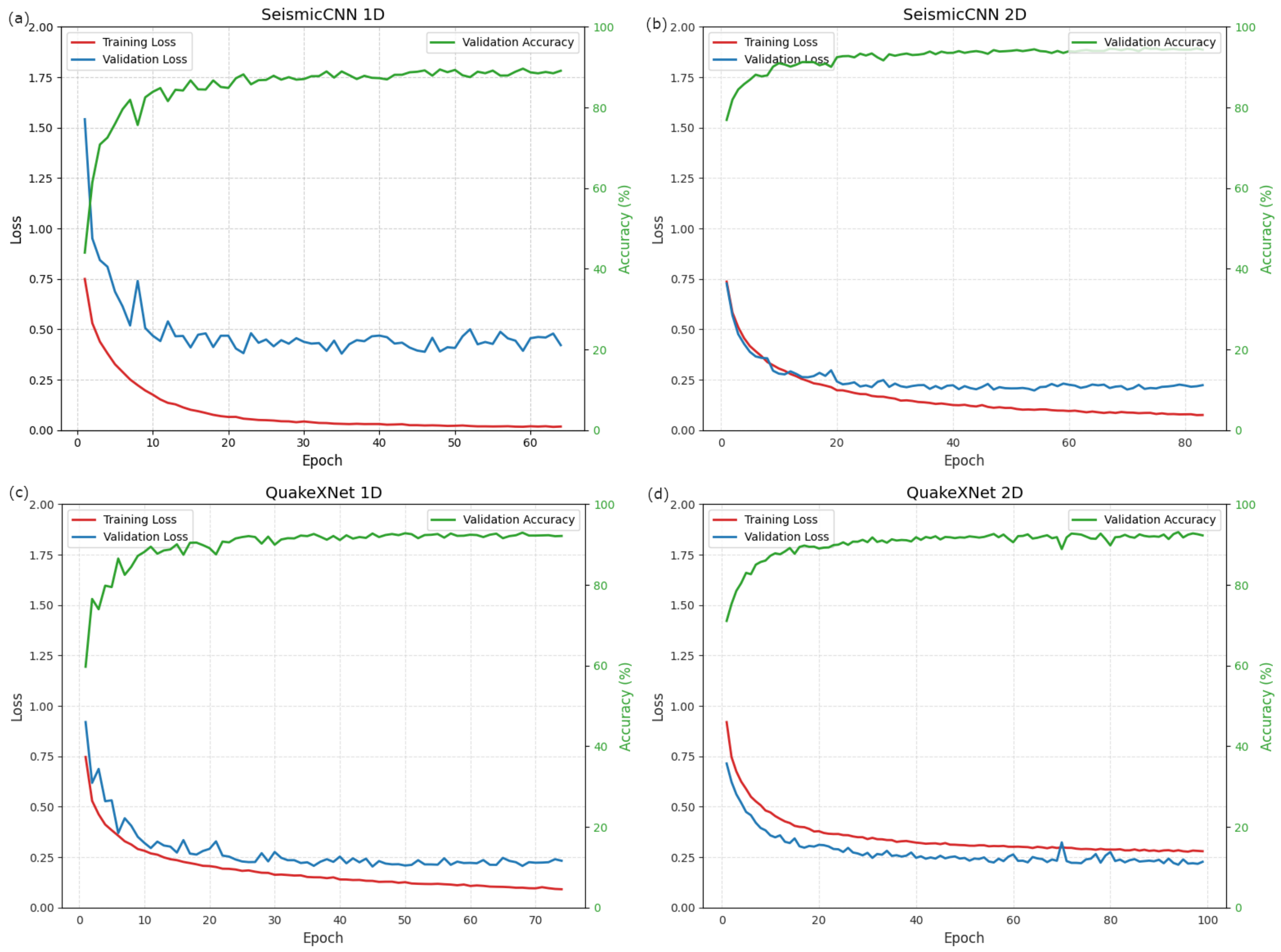

Figure S6 Training and validation accuracy and loss curves for (a) QuakeXNet (1D), (b) QuakeXNet (2D), (c) SeismicCNN (1D) and (d) SeismicCNN (2D).

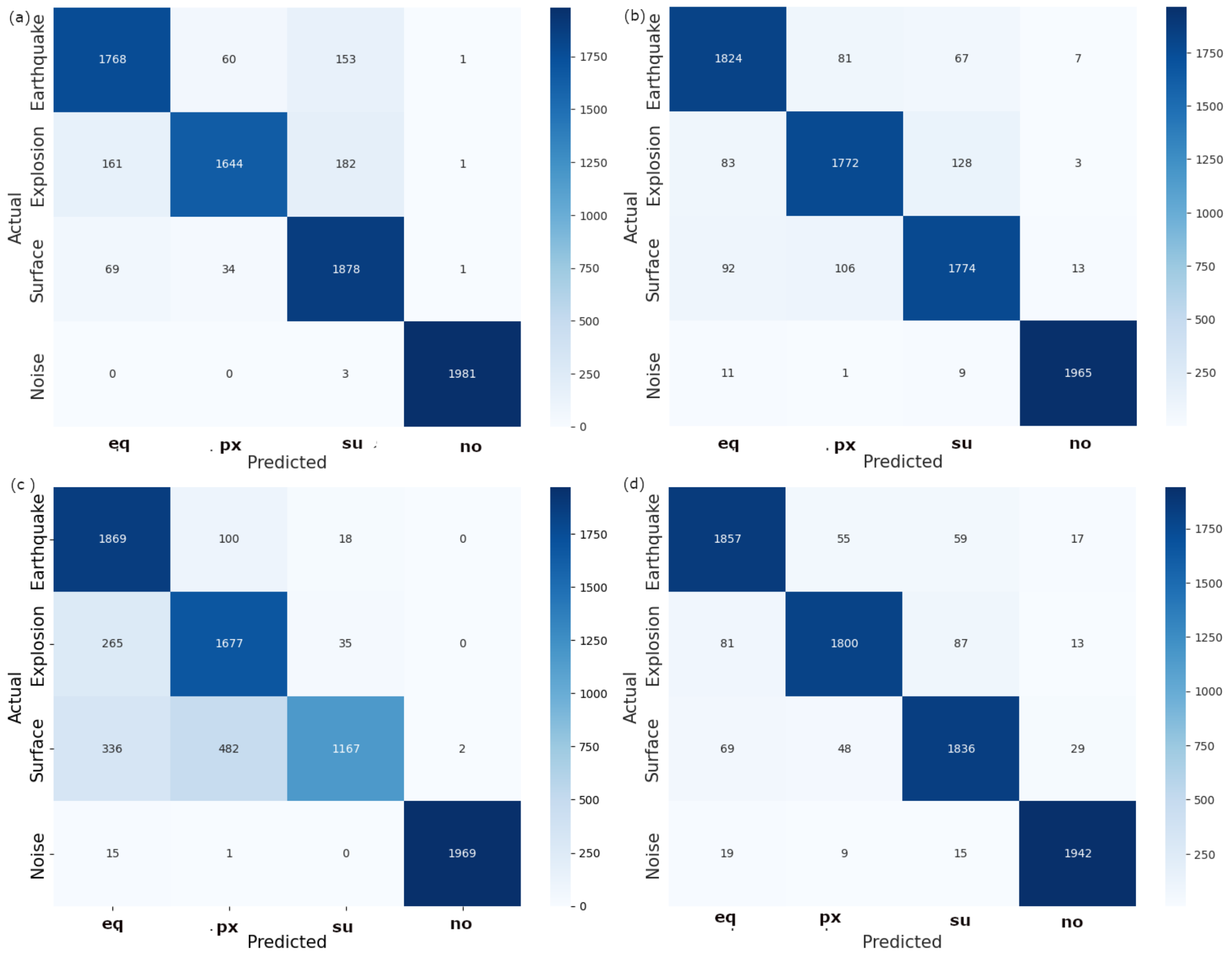

Figure S7 **Confusion Matrix of the four DL models** on the validation set from the curated data sets with (a) QuakeXNet (1D), (b) QuakeXNet (2D), (c) SeismicCNN (1D), (d) SeismicCNN (2D). These confusion matrices highlight where the confusion is among classes.

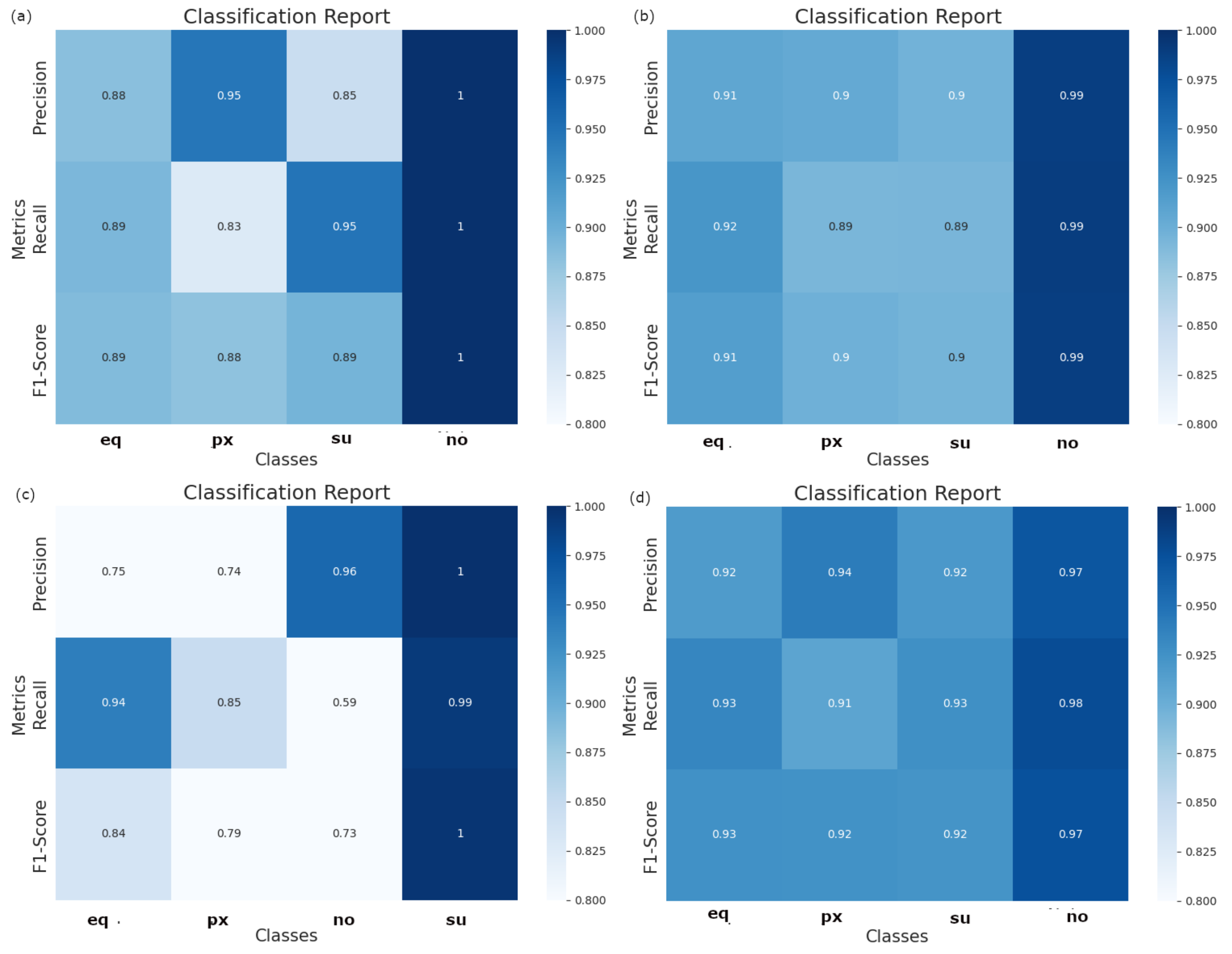

Figure S8 Classification report of the four DL models on the validation set from the curated data sets with (a) QuakeXNet (1D), (b) QuakeXNet (2D), (c) SeismicCNN (1D), (d) SeismicCNN (2D).

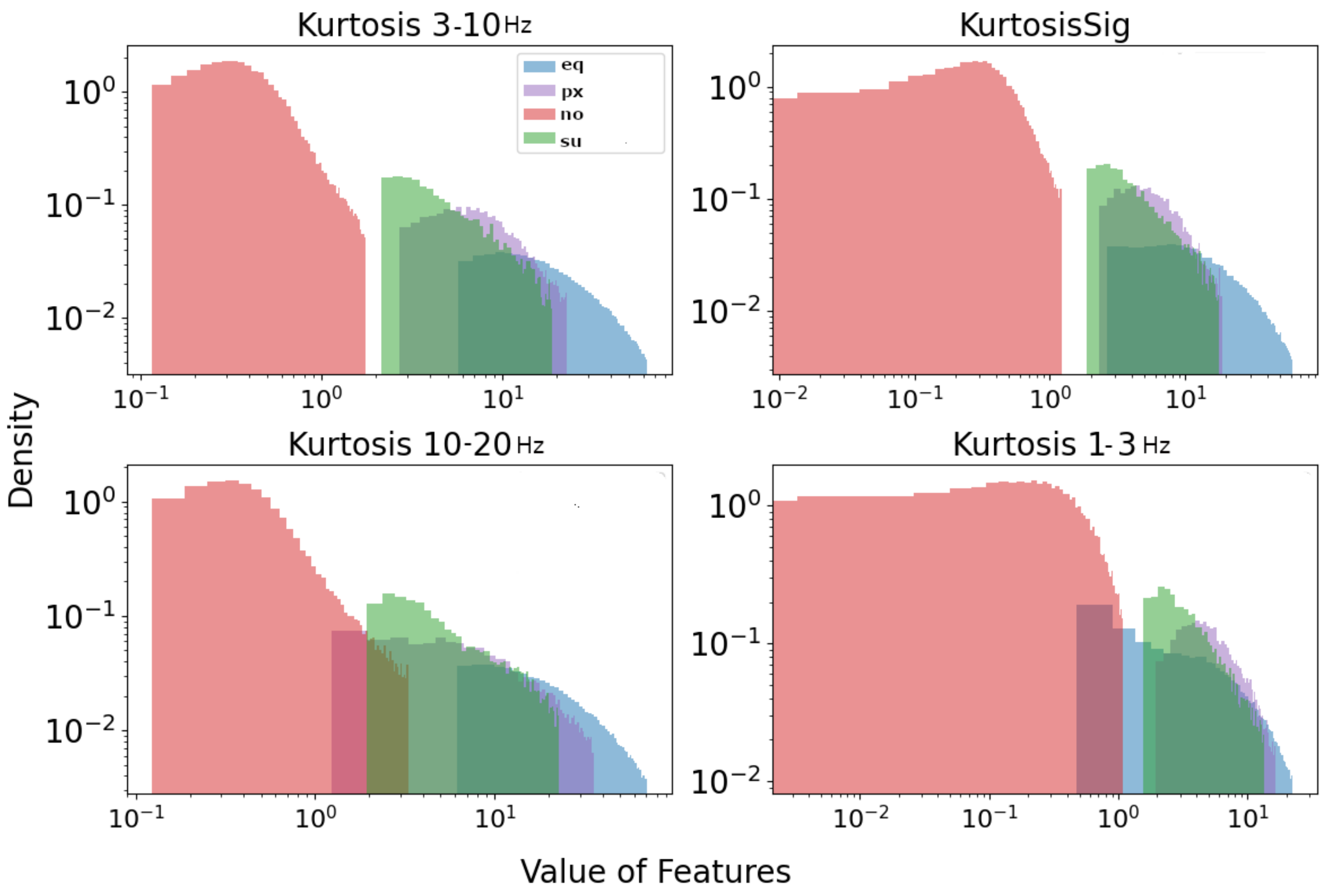

Figure S9 **Histograms of distribution of the most important features** for each class.

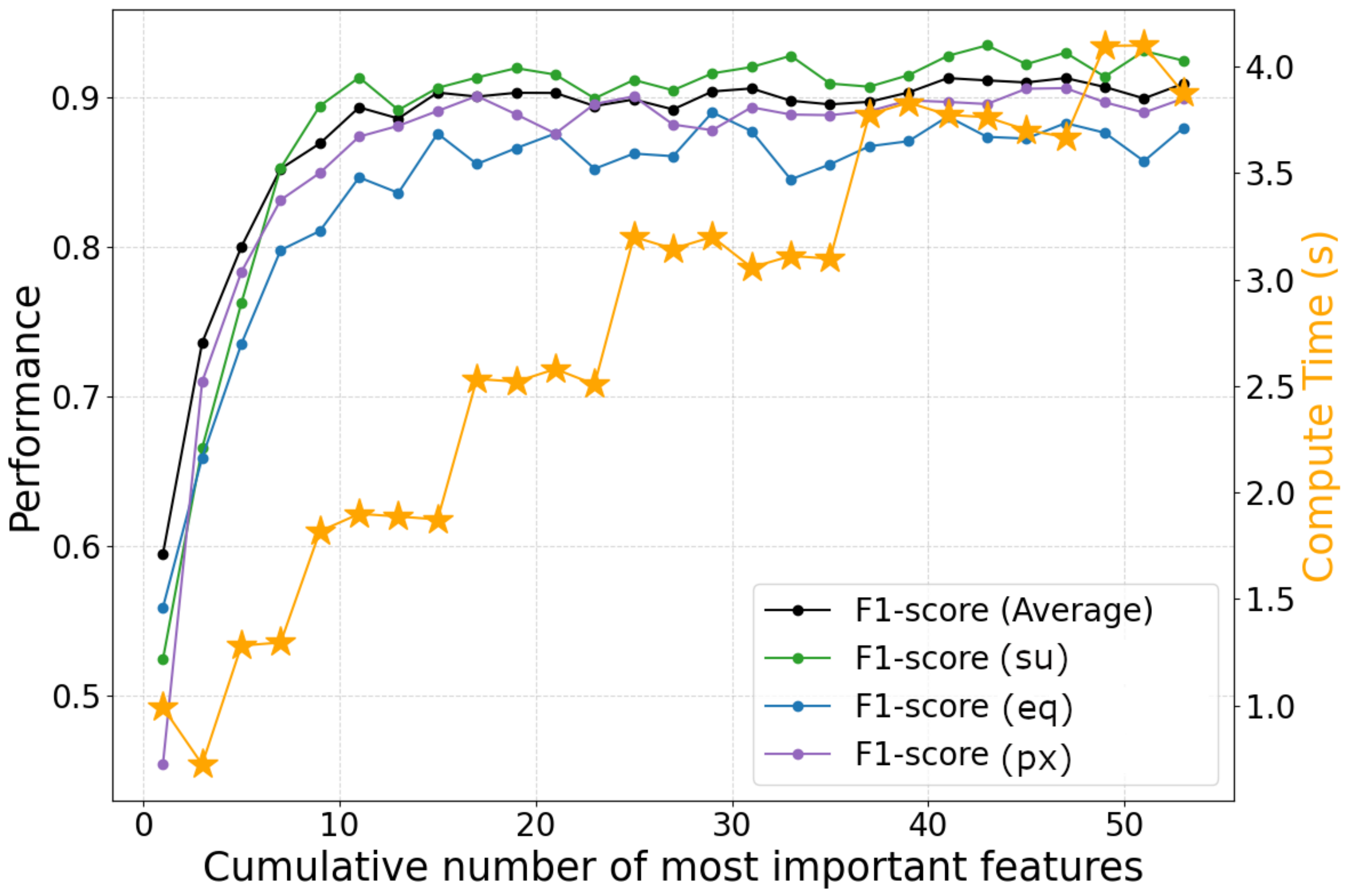

Figure S10 **Performance variation of each class** with cumulatively increasing number of most important features, training time on the twin axis and is indicated in the yellow color.

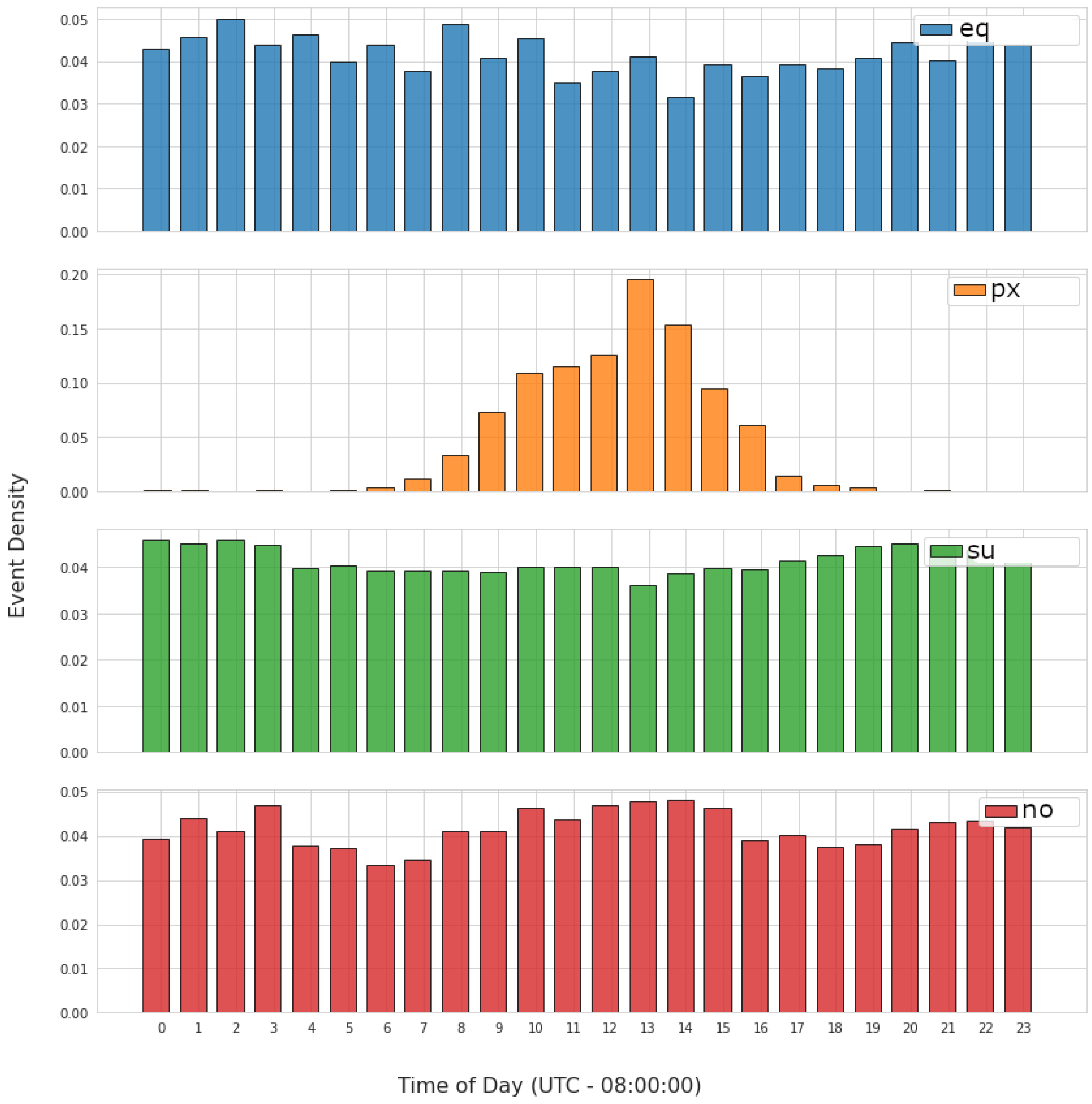

Figure S11 **Hour of the day distributions for all the classes**, this feature was very important for classification.

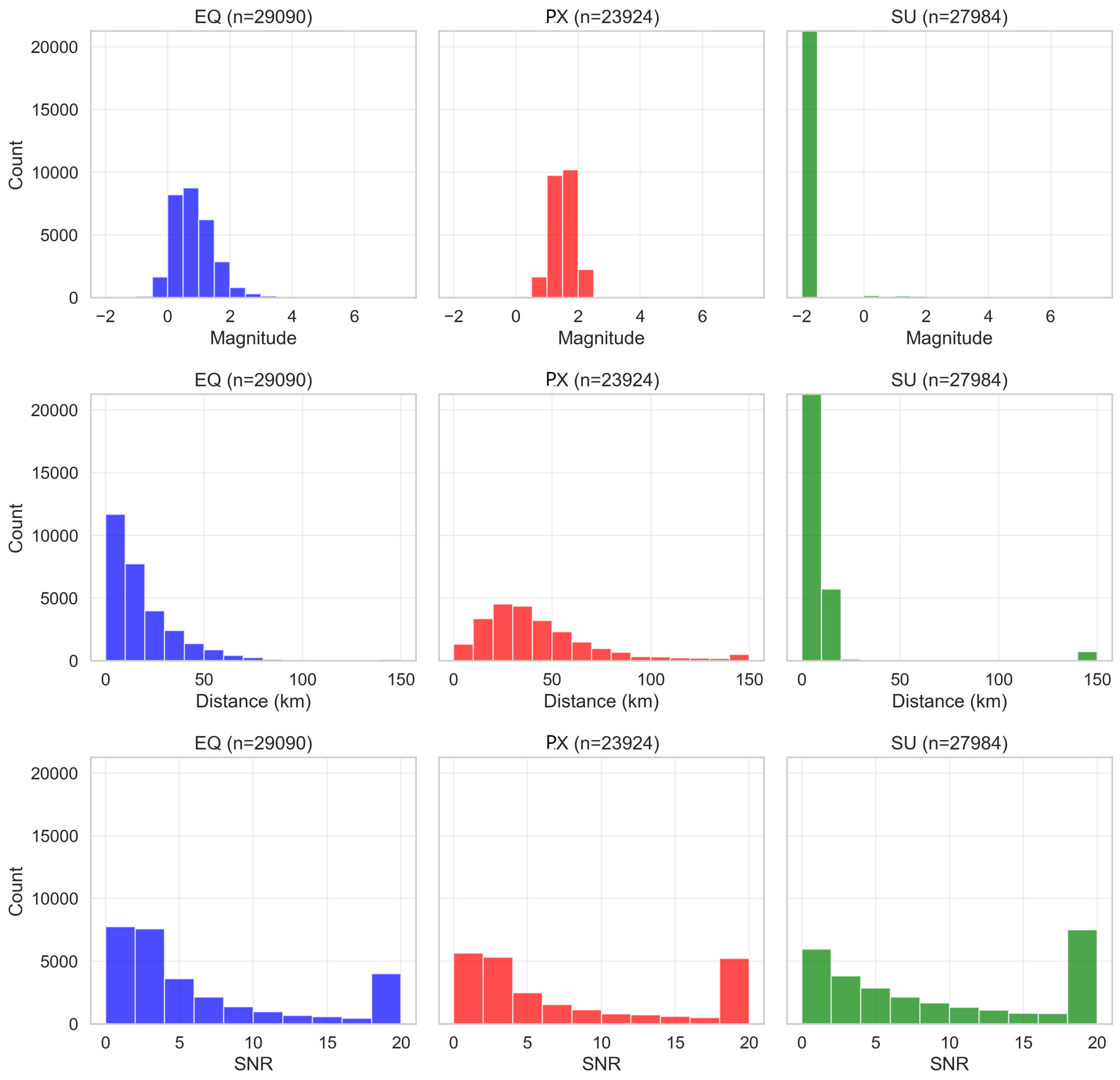

Figure S12 **Distribution of Network Testing data**: with distribution of the magnitudes (top panel), source-receiver range (middle panel), signal-to-noise ratio (bottom panels) and earthquakes (left column and in blue), explosions (middle column and in red), and surface events (in green).

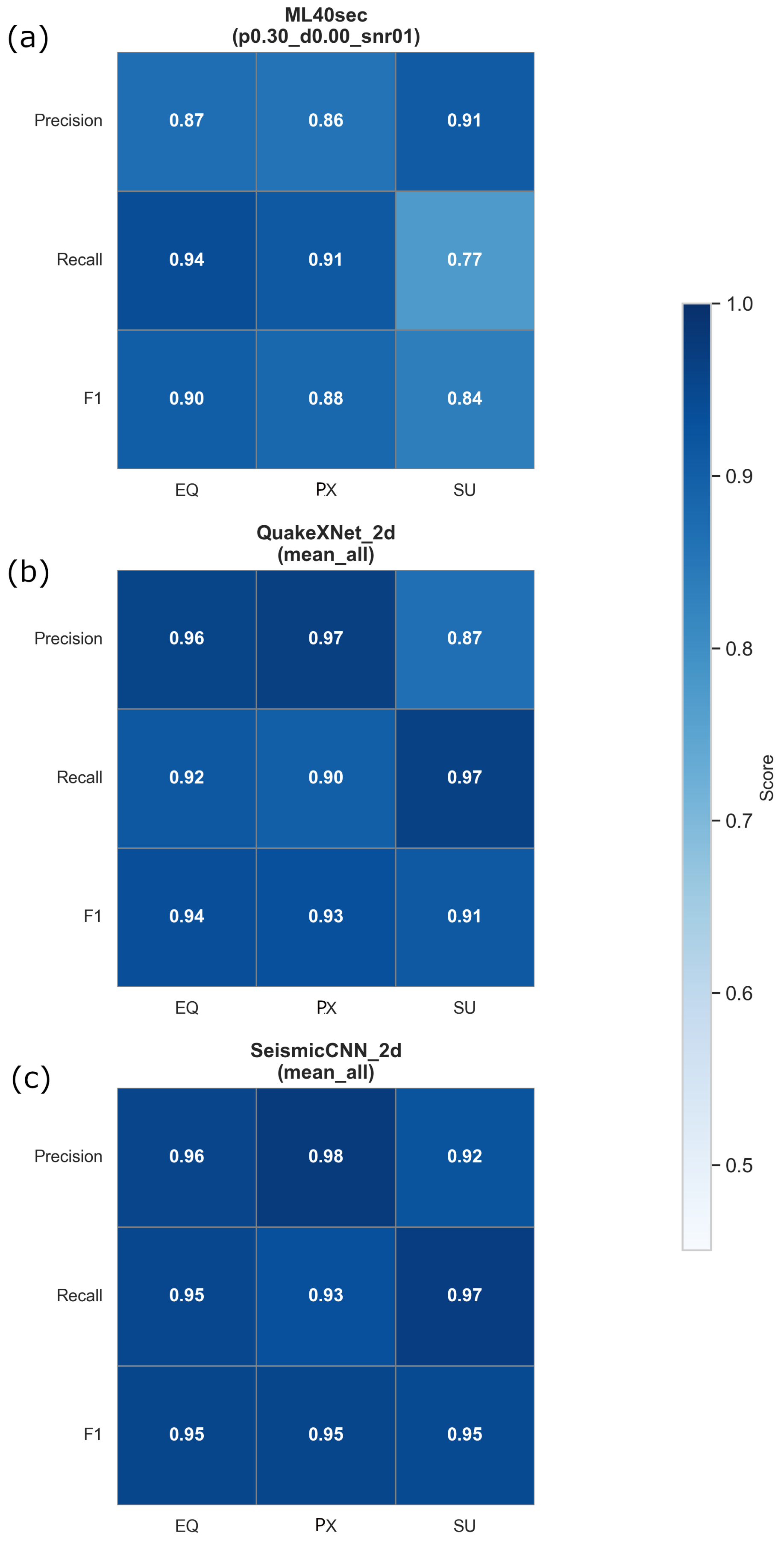

Figure S13 Classificaiton report on the network testing data for three best models: a) ML40sec, b) QuakeXNet_2D, and c) SeismicCNN_2D. These are network-average scores and demonstrate the relative performance among classes on the network test data.

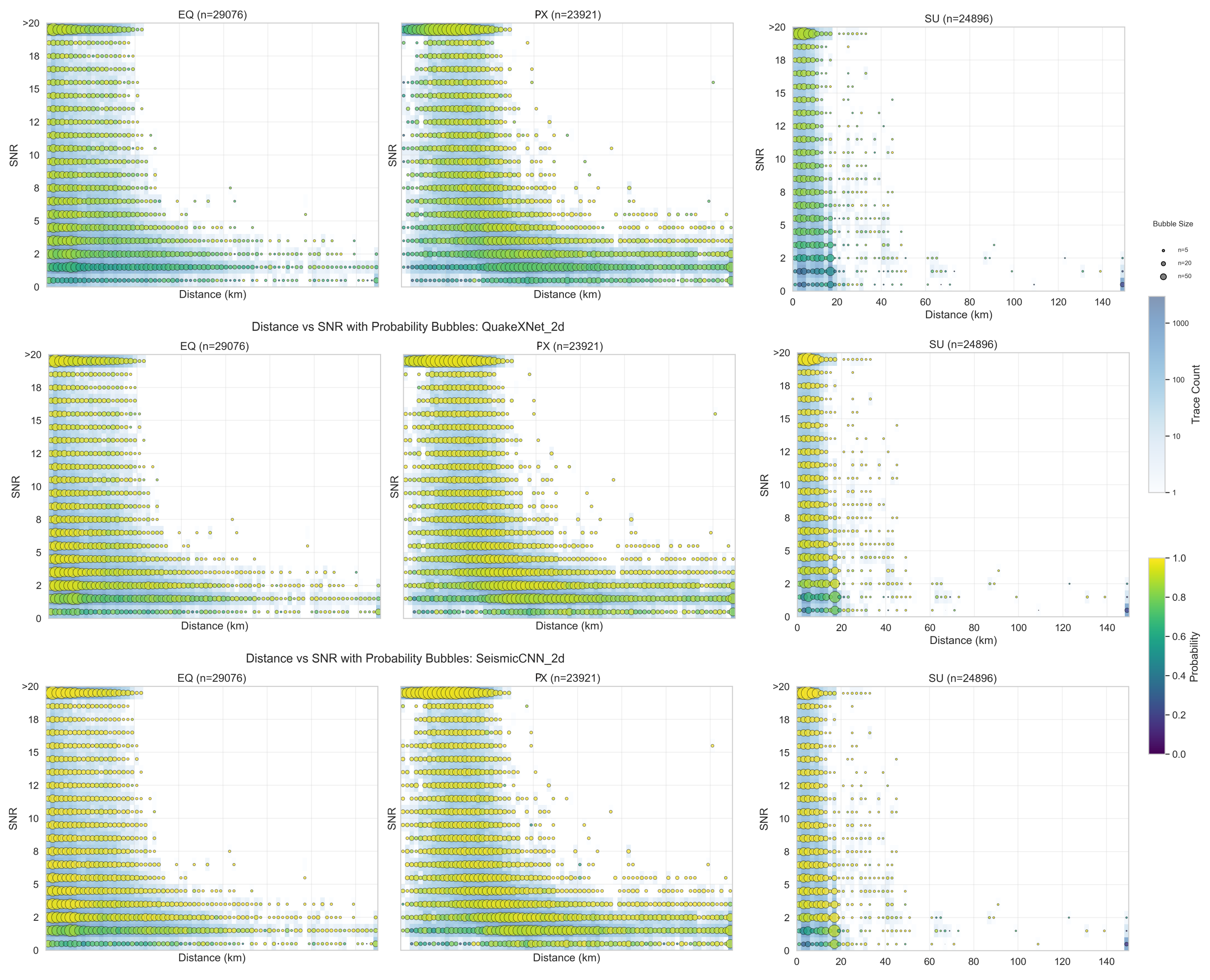

Figure S14 Impacts of SNR and Distance over performance. We show