Vehicle Positioning Using Channel State Information (CSI)

LoS/NLoS Classification and Regression via Machine Learning and Deep Learning

1. Introduction

In this report I present a comprehensive study of vehicle positioning and channel classification using CSI data.
It includes both classification (line-of-sight (LoS)/non-line-of-sight (NLoS)) and regression ((x, y) coordinate prediction) to estimate vehicle position in terms of coordinates using Channel State Information (CSI). These tasks implemented both traditional and deep learning models.
I carried out different preprocessing steps and experimented with Feature reduction techniques through PCA, SparsePCA, KernelPCA with different kernel functions, and Isomap to see the impact of reduced features and how efficiently the variance is getting captured using different PCA methods. These techniques also confirmed nonlinear manifold behavior of the data. Also, various ML algorithms (Logistic Regression, RandomForest, KNN, XGBoost, ANN) were experimented with different regularization and hyperparameter tuning. Both tasks achieved state-of-the-art results for tree-based models and ANN models on validation data, while regression results demonstrated low RMSE and high R^2, confirming the efficacy of the models, validating the suitability of CSI for localization and signal environment characterization.
Note : I have only considered magnitude of the CSI data and discarded the phase information as

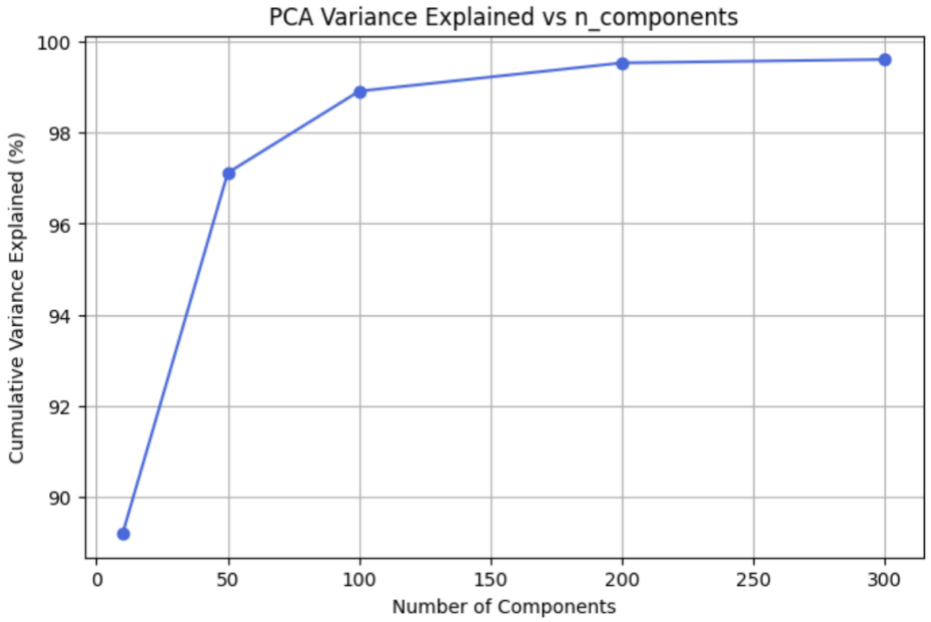
i. Vehicle positioning and LoS/NLoS discrimination depend mainly on power attenuation patterns, not absolute phase.
ii. phase shifts are not consistent across packets or antennas
iii. Reduces feature noise and keeps model training numerically stable
iv. Also, with magnitude only, the models were able to predict with an accuracy of more than 99%.

2. Dataset Description

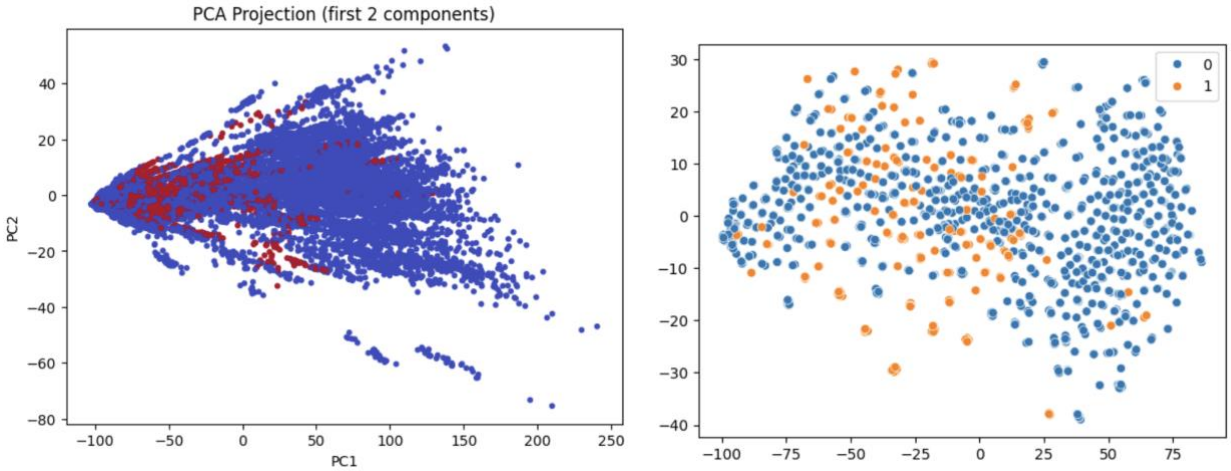
The CSI dataset was recorded from an underground parking lot environment, providing real-world multipath fading effects.
Each sample represents the frequency-domain response of the wireless channel, capturing the propagation characteristics between a vehicle-mounted receiver and a fixed transmitter.
The dataset consists of 15,000 training and 5,000 validation samples, each with 6,528 CSI features extracted from 4 antennas × 1,632 subcarriers (The data was flattened into 6,528-dimensional vectors).
The label vector contains two positional coordinates (x, y) and a binary indicator indication whether the data is line of sight or not(LoS=0, NLoS=1).
Preprocessing involved magnitude computation (which converted complex CSI to absolute amplitude), normalization (standardized each feature with zero mean and unit variance), and PCA-based dimensionality reduction.

3. Dimensionality Reduction & Feature Exploration

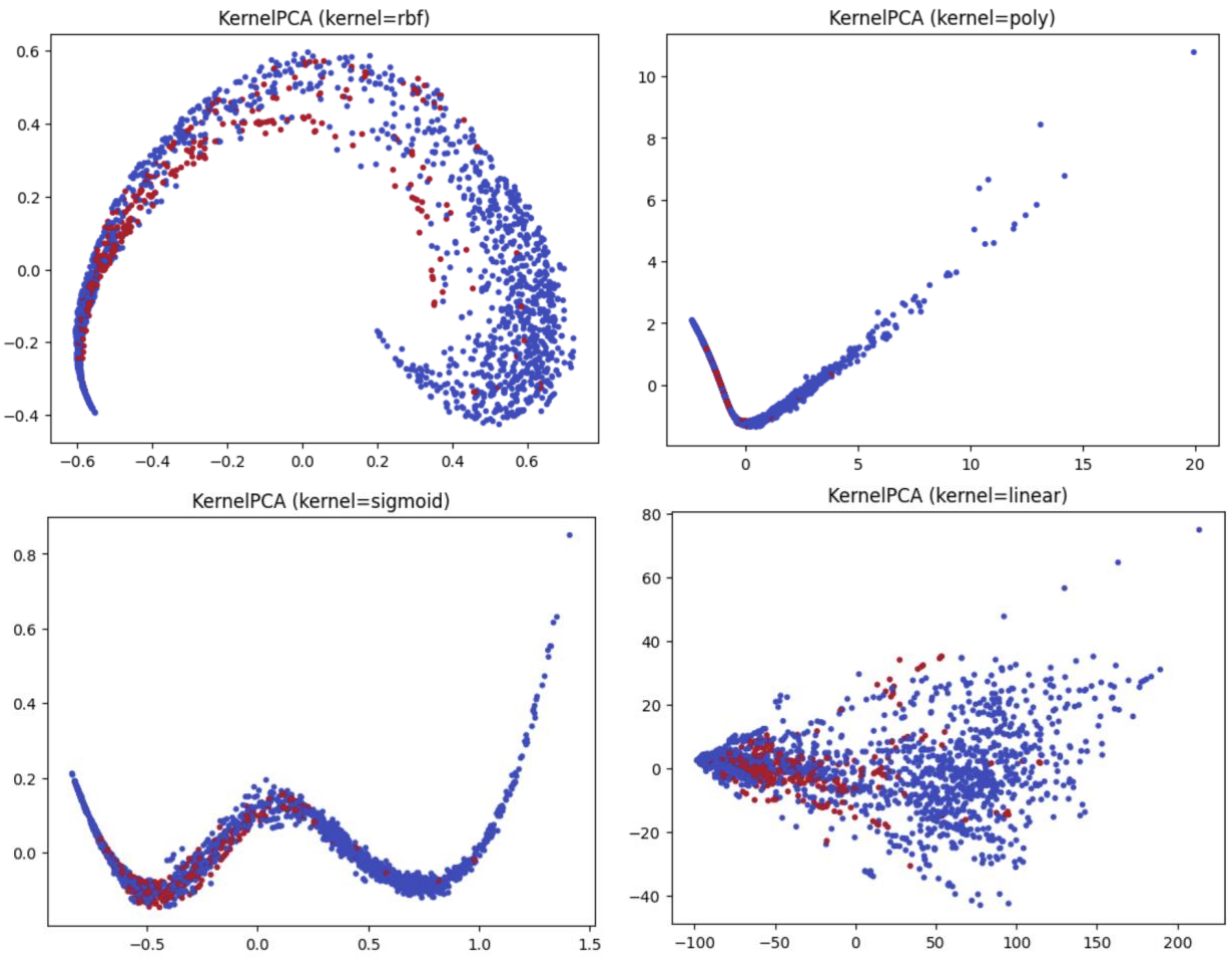
Principal Component Analysis (PCA), SparsePCA, and KernelPCA with kernels like rbf, polynomial and sigmoid were applied to identify low-dimensional manifolds underlying the high-dimensional CSI data.
As shown in the below explained variance vs number of PCA component graph, 100 components capture over 98% of total variance.



Lower dimensions (20–50 components) already captured ~95% variance, confirming redundancy across subcarriers. This ensured efficient computation while preserving positional fidelity.
Also I plotted to see the first 2 component of PCA to see the how the distribution of 2 classes are separate. I employed TSNE on a sample of 2000 datapoints to further confirm high class overlap (as shown in the right plot).



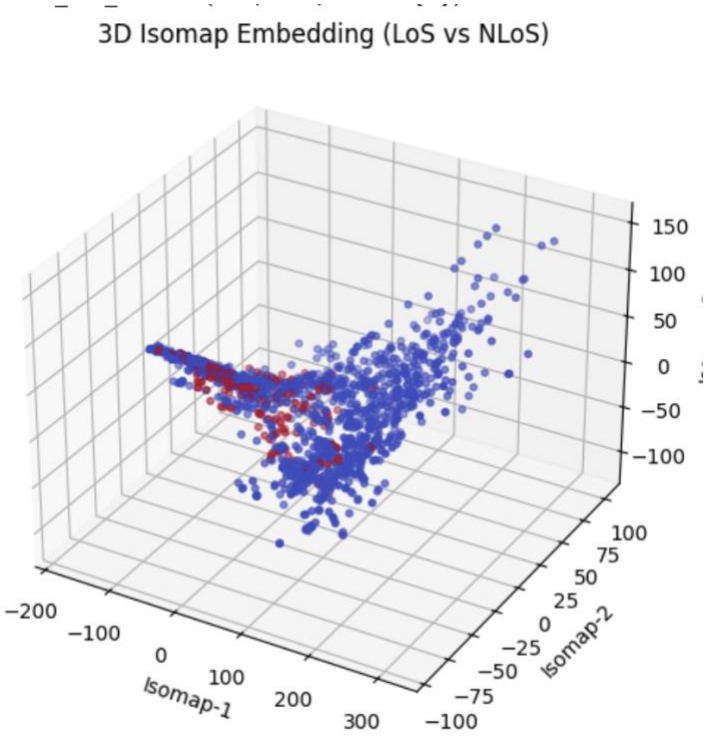
As we can see in the graph, there is lot of overlap between 2 classes for 2 PCA components. Because of this overlap, I tried non-linear PCA methods to see if we are able to segregate the PCs using non-linear kernels.



As seen in above graphs, we are not able to segregate the data in non-ovelapping set for PCs when tried with different kernels.

Isomap manifold embeddings (n_neighbors=10) further confirmed that LoS and NLoS signals form distinct manifolds, reflecting nonlinear geometric separation due to multipath effects.

This also shows CSI amplitude changes smoothly with spatial movement which is ideal for regression which is also confirmed by low Silhouette Score of 0.2821 showing continuous positions.



4. Machine Learning Models & Cross-Validation

4.1 Classification Problem

I started with experimenting different models, The PCA graphs indicates non-linearity of decision boundary, so I kept Logistic Regression with l1 and l2 regularization as base model. Further I experimented with tree based models like Random Forest and Xgboost to capture non-linear boundary in the classification. I also experimented with KNN where I can tweak the complexity of non-linearity of decision boundary by varying number of nearest neighbor.

Cross-validation was performed for Logistic Regression, RandomForest, KNN, and XGBoost classifiers. Also below are the list of hyper-parameters I experimented with.

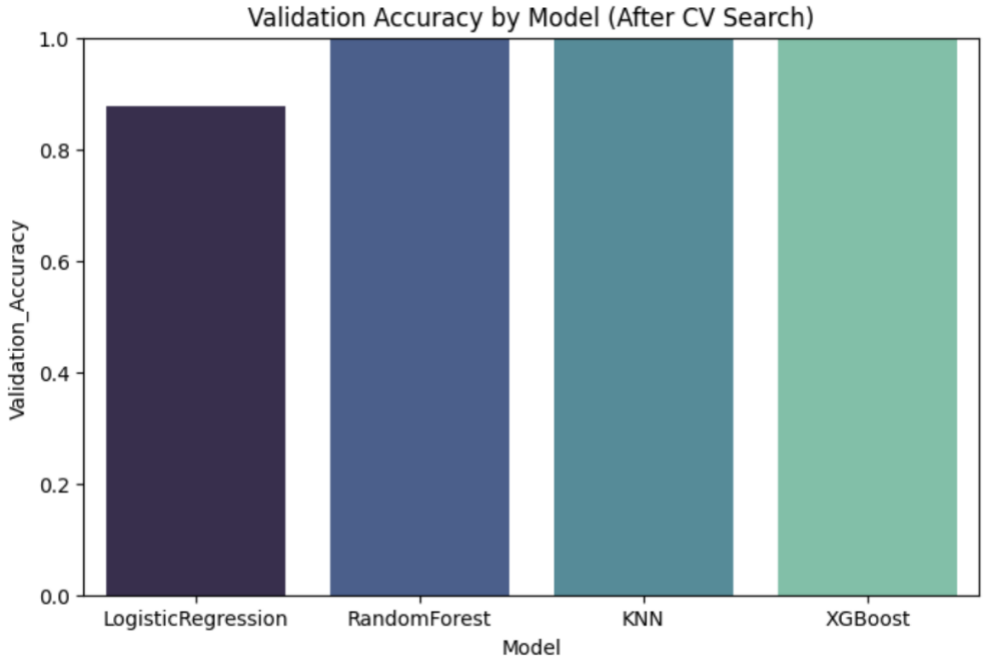
Model	Hyperparameters	Values
LogisticRegression	C	[0.01, 0.1, 1, 10]
	penalty	['l1', 'l2']
	solver	['liblinear']
RandomForest	n_estimators	[50, 100, 200]
	max_depth	[5, 10, None]
KNN	n_neighbors	[3,5,7,9]
	weights	['uniform', 'distance']
XGBoost	n_estimators	[100, 200]

	max_depth	[3, 5, 7]
	learning_rate	[0.01, 0.1, 0.2]
	subsample	[0.8, 1.0]
	colsample_bytree	[0.8, 1.0]

Best hyperparameters were found using GridSearchCV with 3-fold StratifiedKFold splits.
The following summarizes model results:

Model	CV Accuracy	Validation Accuracy	Best Parameters
Logistic Regression	0.883	0.8782	{'C': 0.1, 'penalty': 'l2'}
Random Forest	0.9993	0.9992	{'max_depth': None, 'n_estimators': 200}
KNN	0.9997	0.9994	{'n_neighbors': 3, 'weights': 'uniform'}
XGBoost	0.9998	0.9996	{'max_depth': 7, 'learning_rate': 0.2}

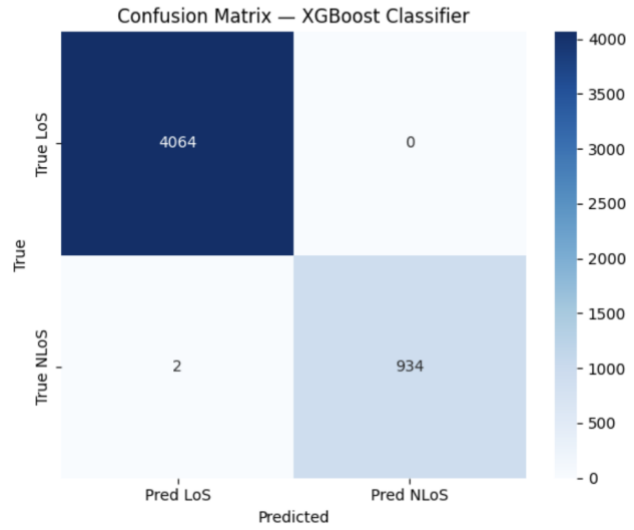
As we can see in the table, we are getting lower validation accuracy for Logistic Regression mainly due to non-linear boundary of the data for LoS and NLoS where as for algorithms with non-linear decision boundary (LR, RF, KNN and XgBoost) are giving me near perfect prediction on validation data.



Below is the confusion metric of Xgboost with best hyper parameters. It shows only 2 out of 5000 datapoints were classified incorrectly.

Validation Accuracy: 0.9996

	precision	recall	f1-score	support
LoS	1.00	1.00	1.00	4064
NLoS	1.00	1.00	1.00	936
accuracy			1.00	5000
macro avg	1.00	1.00	1.00	5000
weighted avg	1.00	1.00	1.00	5000



4.2 Regression Problem

I started with experimenting different models for Regression Problem as well, I kept Linear Regression as base model. The results that I got from Linear Regression were not promising so I moved to other regularized and non-linear complex models.

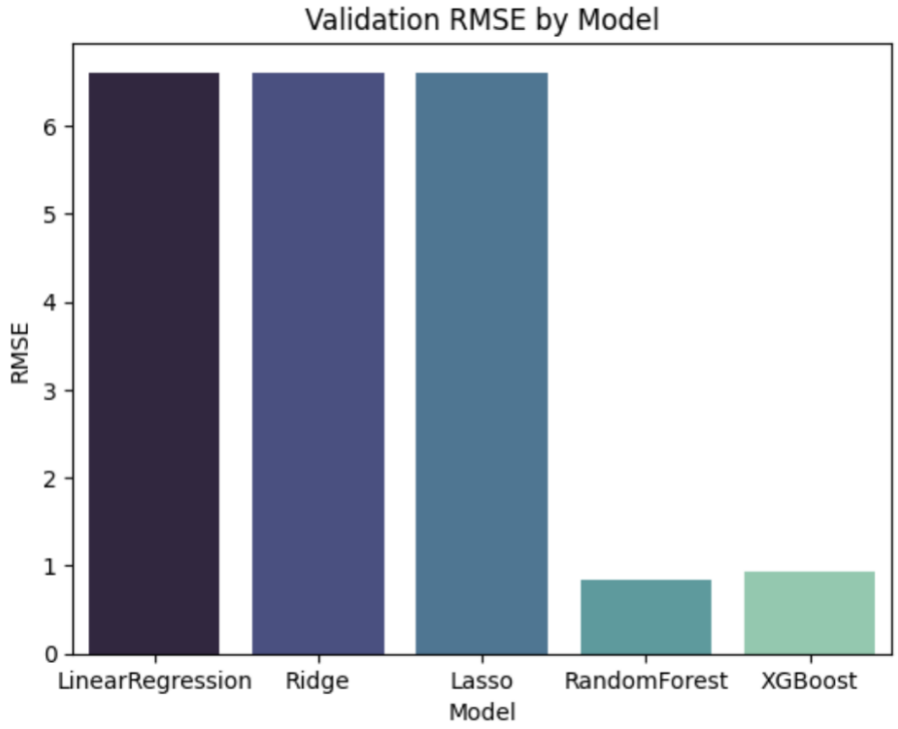
Below are the list of models and hyper-parameters that I experimented with:

Model	Hyperparameters (Grid Search Range)	Estimator / Configuration
Linear Regression	{}	LinearRegression()
Ridge Regression	alpha: [0.01, 0.1, 1, 10]	Ridge()
Lasso Regression	alpha: [0.001, 0.01, 0.1, 1]	Lasso()
Random Forest Regressor	n_estimators: [100, 200] max_depth: [5, 10, None]	RandomForestRegressor(random_state=42)
XGBoost Regressor	n_estimators: [100, 200] max_depth: [3, 5, 7] learning_rate: [0.01, 0.1, 0.2]subsample: [0.8, 1.0]	XGBRegressor(objective='reg:squarederror', eval_metric='rmse', tree_method='gpu_hist' if torch.cuda.is_available() else 'auto', random_state=42)

Out of this I got following results for best hyperparameters for each of the model.

Model	RMSE	R^2	Best Parameters
Linear Regression	6.6042	0.5391	{}
Ridge Regression	6.6042	0.5391	{'alpha': 10}
Lasso Regression	6.6045	0.5391	{'alpha': 0.01}
Random Forest	0.8411	0.9929	{'max_depth': None, 'n_estimators': 200}
XGBoost	0.929	0.9913	{'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 200, 'subsample': 0.8}

As we can see R-square is low for regression based models with regularization, whereas tree-based model has high R-square and low RMSE on validation data which shows the tree based algorithms are better able to map the complexity of the data.



5. Deep Learning Architecture (ANN)

5.1 Classification Problem

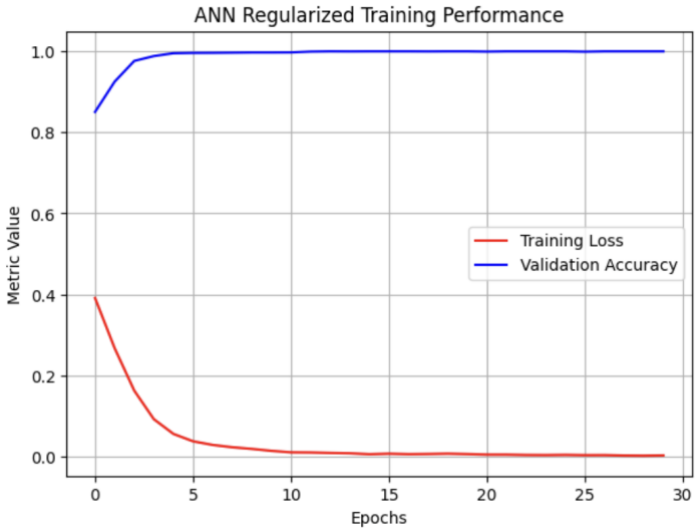
The ANN model comprised three fully connected layers with ReLU activations and dropout regularization to prevent overfitting.
It was trained using the Adam optimizer with a learning rate of 0.001 and weight decay of 1e-4.
The network architecture is as follows:

Input Layer: 100 PCA features
Hidden Layer 1: 128 neurons, ReLU + Dropout(0.3)
Hidden Layer 2: 64 neurons, ReLU + Dropout(0.3)
Output Layer: 1 neuron, Sigmoid activation (binary classification)

Loss Function: Binary Cross Entropy
Optimizer: Adam (lr=0.001, weight_decay=1e-4)

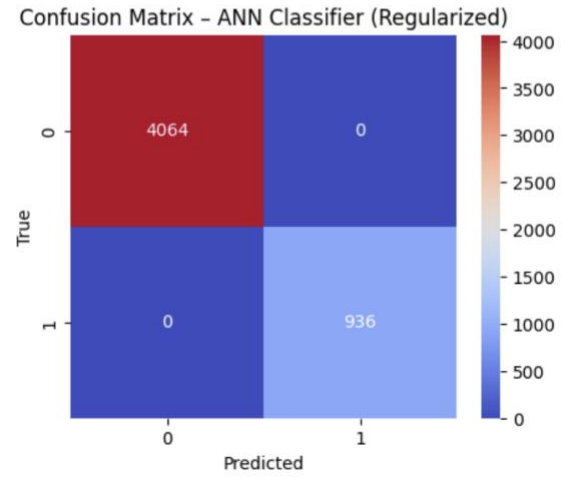
As shown in the epoch log, training converged rapidly, with validation accuracy reaching 100% within 13 epochs. Also as shown in the below plot training accuracy is converging very rapidly i.e. around 4-5 epoch and validation error converging around 13 epoch.

```
=== ANN (with Dropout & L2 Regularization) ===
Epoch 01/30 | Loss=0.3908 | ValAcc=0.8504
Epoch 02/30 | Loss=0.2669 | ValAcc=0.9252
Epoch 03/30 | Loss=0.1621 | ValAcc=0.9764
Epoch 04/30 | Loss=0.0915 | ValAcc=0.9884
Epoch 05/30 | Loss=0.0551 | ValAcc=0.9954
Epoch 06/30 | Loss=0.0369 | ValAcc=0.9962
Epoch 07/30 | Loss=0.0279 | ValAcc=0.9964
Epoch 08/30 | Loss=0.0222 | ValAcc=0.9968
Epoch 09/30 | Loss=0.0181 | ValAcc=0.9972
Epoch 10/30 | Loss=0.0132 | ValAcc=0.9972
Epoch 11/30 | Loss=0.0096 | ValAcc=0.9974
Epoch 12/30 | Loss=0.0092 | ValAcc=0.9994
Epoch 13/30 | Loss=0.0081 | ValAcc=1.0000
Epoch 14/30 | Loss=0.0073 | ValAcc=0.9998
Epoch 15/30 | Loss=0.0049 | ValAcc=1.0000
Epoch 16/30 | Loss=0.0063 | ValAcc=1.0000
Epoch 17/30 | Loss=0.0051 | ValAcc=1.0000
Epoch 18/30 | Loss=0.0056 | ValAcc=0.9998
Epoch 19/30 | Loss=0.0064 | ValAcc=1.0000
Epoch 20/30 | Loss=0.0053 | ValAcc=1.0000
Epoch 21/30 | Loss=0.0040 | ValAcc=0.9994
Epoch 22/30 | Loss=0.0040 | ValAcc=1.0000
Epoch 23/30 | Loss=0.0031 | ValAcc=1.0000
Epoch 24/30 | Loss=0.0029 | ValAcc=1.0000
Epoch 25/30 | Loss=0.0034 | ValAcc=1.0000
Epoch 26/30 | Loss=0.0027 | ValAcc=0.9992
Epoch 27/30 | Loss=0.0028 | ValAcc=1.0000
Epoch 28/30 | Loss=0.0017 | ValAcc=1.0000
Epoch 29/30 | Loss=0.0013 | ValAcc=1.0000
Epoch 30/30 | Loss=0.0020 | ValAcc=1.0000
```



The confusion matrix was perfectly diagonal, confirming flawless LoS/NLoS separation.

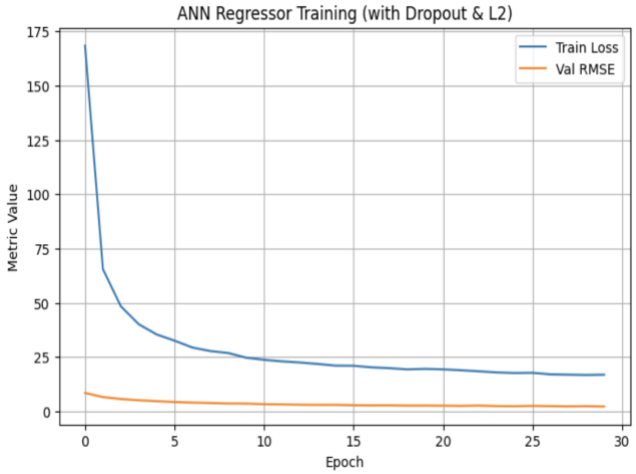
```
=== FINAL ANN RESULTS ===
Validation Accuracy: 1.0000
Confusion Matrix:
 [[4064    0]
 [   0  936]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4064
           1       1.00      1.00      1.00       936

    accuracy                           1.00      5000
   macro avg       1.00      1.00      1.00      5000
weighted avg       1.00      1.00      1.00      5000
```

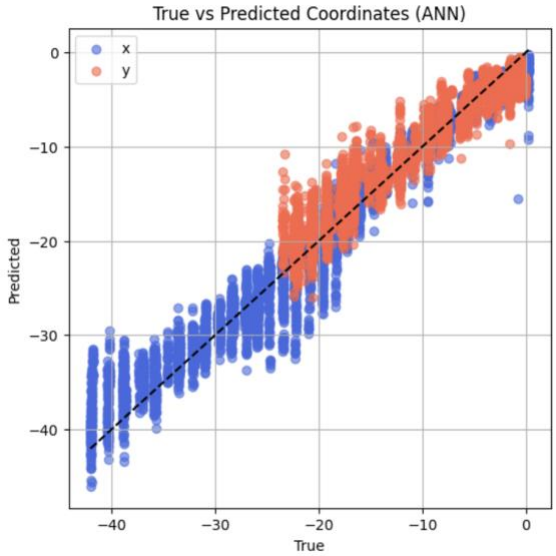


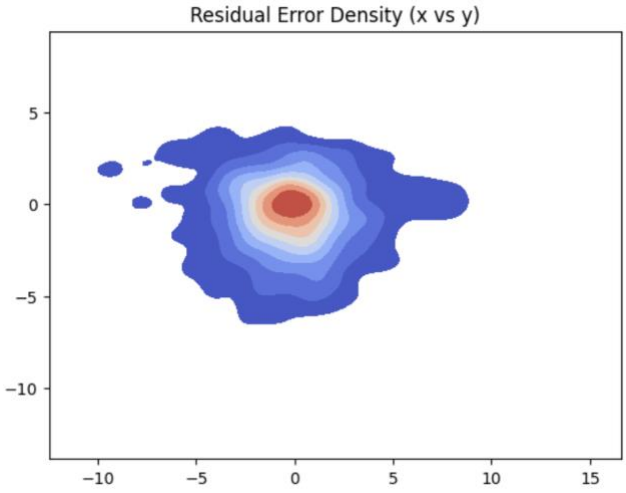
5.2 Regression Problem

Experiment 1 (with the similar architecture as in classification problem)
Input Layer: 100 PCA features
Hidden Layer 1: 128 neurons, ReLU + Dropout(0.3)
Hidden Layer 2: 64 neurons, ReLU + Dropout(0.3)
Output Layer: 2 neuron (x and y coordinates)
Loss Function: MSE Loss
Optimizer: Adam (lr=0.001, weight_decay=1e-4)
ANN Final RMSE=2.2530, R^2=0.9412

```
Epoch 01/30 | Loss=168.4046 | ValRMSE=8.4790
Epoch 02/30 | Loss=65.5012 | ValRMSE=6.5841
Epoch 03/30 | Loss=48.4072 | ValRMSE=5.7092
Epoch 04/30 | Loss=40.1425 | ValRMSE=5.1050
Epoch 05/30 | Loss=35.4312 | ValRMSE=4.7062
Epoch 06/30 | Loss=32.5742 | ValRMSE=4.3155
Epoch 07/30 | Loss=29.4079 | ValRMSE=4.0276
Epoch 08/30 | Loss=27.7806 | ValRMSE=3.8620
Epoch 09/30 | Loss=26.8864 | ValRMSE=3.6295
Epoch 10/30 | Loss=24.7319 | ValRMSE=3.5904
Epoch 11/30 | Loss=23.7520 | ValRMSE=3.3083
Epoch 12/30 | Loss=23.0504 | ValRMSE=3.2130
Epoch 13/30 | Loss=22.5161 | ValRMSE=3.0793
Epoch 14/30 | Loss=21.8534 | ValRMSE=3.0007
Epoch 15/30 | Loss=21.0734 | ValRMSE=3.0237
Epoch 16/30 | Loss=21.0222 | ValRMSE=2.8473
Epoch 17/30 | Loss=20.2924 | ValRMSE=2.7651
Epoch 18/30 | Loss=19.9134 | ValRMSE=2.7844
Epoch 19/30 | Loss=19.3670 | ValRMSE=2.6792
Epoch 20/30 | Loss=19.5835 | ValRMSE=2.6783
Epoch 21/30 | Loss=19.3656 | ValRMSE=2.6237
Epoch 22/30 | Loss=18.9535 | ValRMSE=2.5274
Epoch 23/30 | Loss=18.4652 | ValRMSE=2.6845
Epoch 24/30 | Loss=17.9549 | ValRMSE=2.4390
Epoch 25/30 | Loss=17.6789 | ValRMSE=2.3846
Epoch 26/30 | Loss=17.7722 | ValRMSE=2.5247
Epoch 27/30 | Loss=17.0651 | ValRMSE=2.4204
Epoch 28/30 | Loss=16.9245 | ValRMSE=2.3108
Epoch 29/30 | Loss=16.7782 | ValRMSE=2.3835
Epoch 30/30 | Loss=16.8910 | ValRMSE=2.2530
```







Above scatter plots showed true vs. predicted coordinates following the y=x diagonal, and residual error density plots were Gaussian-shaped, centered at zero, indicating unbiased regression mapping.

The R^2 gap (~94% ANN vs 99% XGBoost) means my ANN is underfitting, suggesting the model isn't expressive enough to capture the nonlinear interactions in the high-dimensional CSI features.

We would fix that by increasing depth, width, normalization, and regularization control in a balanced way to avoid overfitting.

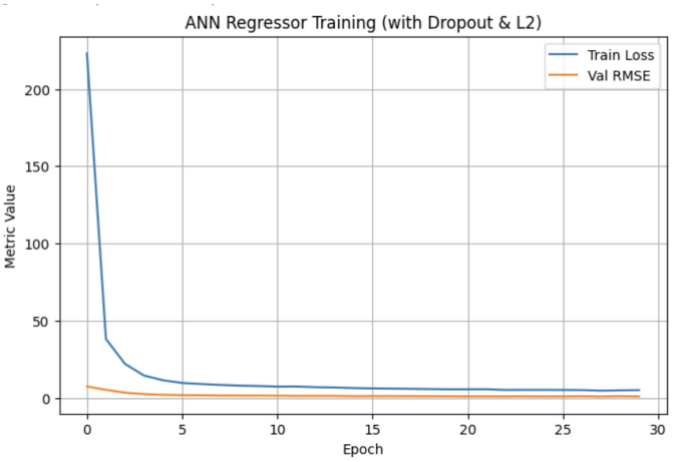
Experiment 2 (with more dense architecture as compared with experiment 1)
Input Layer: 100 PCA features
Hidden Layer 1: 256 neurons, Batch Normalization, ReLU + Dropout(0.3)
Hidden Layer 2: 128 neurons, Batch Normalization, ReLU + Dropout(0.3)
Hidden Layer 3: 64 neurons, ReLU
Output Layer: 2 neuron (x and y coordinates)
Loss Function: MSE Loss
Optimizer: Adam (lr=0.001, weight_decay=1e-4)
ANN Final RMSE=1.4583, R^2=0.9778

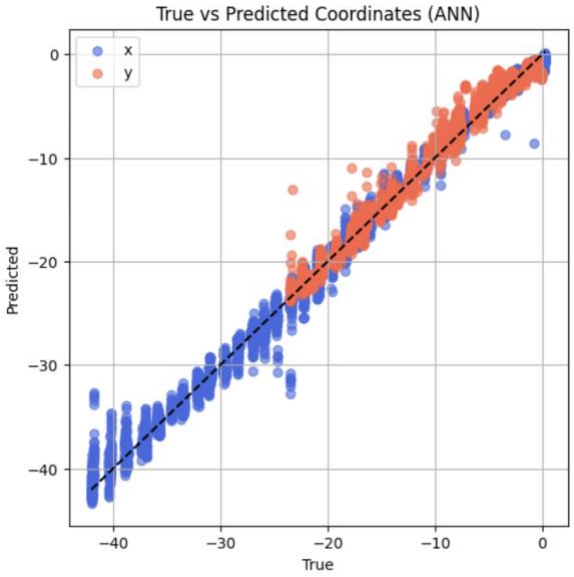
Experiment 3 (with further denser architecture as compared with experiment 2)
Input Layer: 100 PCA features
Hidden Layer 1: 512 neurons, Batch Normalization, ReLU + Dropout(0.25)
Hidden Layer 2: 256 neurons, Batch Normalization, ReLU + Dropout(0.3)
Hidden Layer 3: 128 neurons, Batch Normalization, ReLU + Dropout(0.3)
Hidden Layer 4: 64 neurons, ReLU
Output Layer: 2 neuron (x and y coordinates)
Loss Function: MSE Loss
Optimizer: Adam (lr=0.001, weight_decay=1e-4)
ANN Final RMSE=0.9361, R^2=0.9902

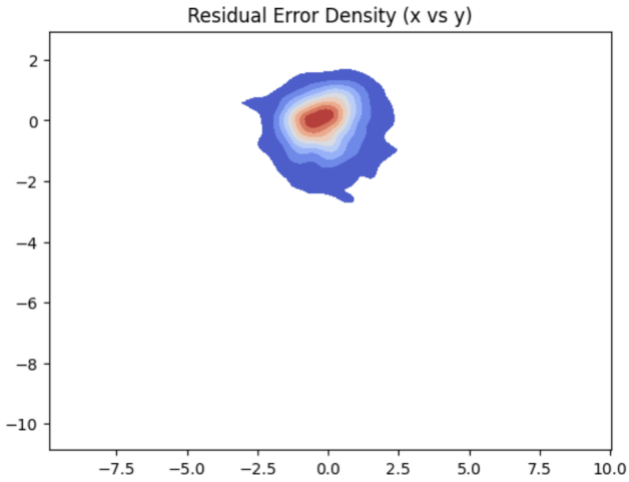
Enhancement in the architecture as compared with the Experiment-1

Improvement	Reasoning
Wider early layers (512→256)	Captures complex nonlinear combinations in high-dimensional PCA/CSI features.
Batch Normalization	Stabilizes training, reduces internal covariate shift, enables higher learning rates.
Deeper network (4 hidden layers)	Increases representational power to approximate nonlinear spatial mappings.
Dropout tuning (0.25–0.3)	Reduces overfitting but keeps enough neurons active for learning high variance signals.
ReLU throughout	Maintains nonlinearity and fast convergence.

```
Epoch 01/30 | Loss=223.0886 | ValRMSE=7.3869
Epoch 02/30 | Loss=38.1155 | ValRMSE=5.2032
Epoch 03/30 | Loss=21.9954 | ValRMSE=3.3338
Epoch 04/30 | Loss=14.4293 | ValRMSE=2.4319
Epoch 05/30 | Loss=11.3832 | ValRMSE=1.9939
Epoch 06/30 | Loss=9.6590 | ValRMSE=1.7583
Epoch 07/30 | Loss=8.9612 | ValRMSE=1.6765
Epoch 08/30 | Loss=8.3937 | ValRMSE=1.5292
Epoch 09/30 | Loss=7.9252 | ValRMSE=1.5058
Epoch 10/30 | Loss=7.6614 | ValRMSE=1.4796
Epoch 11/30 | Loss=7.2985 | ValRMSE=1.3950
Epoch 12/30 | Loss=7.3575 | ValRMSE=1.2535
Epoch 13/30 | Loss=6.9181 | ValRMSE=1.2848
Epoch 14/30 | Loss=6.7509 | ValRMSE=1.2659
Epoch 15/30 | Loss=6.3386 | ValRMSE=1.1324
Epoch 16/30 | Loss=6.1306 | ValRMSE=1.1298
Epoch 17/30 | Loss=6.0028 | ValRMSE=1.1404
Epoch 18/30 | Loss=5.8567 | ValRMSE=1.1112
Epoch 19/30 | Loss=5.6917 | ValRMSE=1.0706
Epoch 20/30 | Loss=5.5340 | ValRMSE=1.0282
Epoch 21/30 | Loss=5.5529 | ValRMSE=0.9899
Epoch 22/30 | Loss=5.5761 | ValRMSE=1.0153
Epoch 23/30 | Loss=5.1226 | ValRMSE=0.9516
Epoch 24/30 | Loss=5.1891 | ValRMSE=1.0165
Epoch 25/30 | Loss=5.1724 | ValRMSE=0.9334
Epoch 26/30 | Loss=5.1360 | ValRMSE=0.9648
Epoch 27/30 | Loss=5.0381 | ValRMSE=1.0468
Epoch 28/30 | Loss=4.6569 | ValRMSE=0.8743
Epoch 29/30 | Loss=4.8677 | ValRMSE=1.0666
Epoch 30/30 | Loss=5.0079 | ValRMSE=0.9361
```







As shown in the loss vs epoch curve, both training and validation loss converges rapidly i.e around 5 epoch, also the stable convergence.
Scatter plots showed true vs. predicted coordinates closely following the y=x diagonal, and residual error density plots were Gaussian-shaped, centered at zero, indicating unbiased regression mapping, both the graphs have much lower variance as depicted in the similar graphs for experiment 1. Also the model evaluation metrics are near perfect for R-square and very low for RMSE, bridging the performance gap that we observed in experiment 1.

7. Comparative Results & Insights

All advanced models achieved near-perfect performance.
Tree-based and ANN architectures excelled due to their capacity to capture nonlinear relationships in multipath CSI data.
Linear models struggled because of inherent nonlinearity in signal propagation.
The following observations were made:
- PCA effectively reduced noise while preserving key variance.
- KernelPCA and Isomap confirmed nonlinear feature manifolds.
- XGBoost achieved balance between interpretability and accuracy.
- ANN showed rapid convergence and excellent regularization behavior.
- Residuals confirmed minimal bias, validating spatial model consistency.

8. Conclusion

This project demonstrates the effectiveness of ML and DL models for CSI-based vehicle positioning.
ANN and XGBoost models outperformed all others, achieving 99.9–100% classification accuracy and sub-decimeter regression RMSE.
Dimensionality reduction using PCA ensured computational efficiency without performance loss.

The data and prediction accuracy did not require us to explore more advanced models such as convolutional architectures (CNNs) and attention-based models.