GROUP 4 – PROJECT B: COMPARISON OF NOVEL APPROACHES TO IMPROVE THE PERFORMANCE OF MODEL FOR PREDICTING IMPACT LOCATION

Rushikesh Shende* (415551), Soji Jacob* (416137), Renuka Misal* (416138), Pratik Tatipamula* (429272), Soham Bhute* (416002)

***M.Sc. Computer Aided Mechanical Engineering, RWTH Aachen University**

Bad weather conditions during an aircraft flight pose a severe threat to the integrity of the aircraft structure. A correct methodology to gauge the location and severity of the impact can be useful in making decisions post-impact. The goal of the present study is to improve upon the Convolution Neural Network model developed previously by hyper-parameter tuning and sensor coupling. Artificial data augmentation is implemented by adding a 'controlled noise' to the input. In addition, Ensemble learning methods (Random Forests), Classification CNN and separate networks for X and Y coordinates are also explored. The applicability of generative models such as GANs and Auto Encoders are also discussed.

KEYWORDS: Convolution Neural Networks, Controlled Noise, Ensemble Learning Methods, Hyperparameter Tuning, Classification Network

INTRODUCTION:

In the present study [1], a small-scale idealization of the impact is done using a ball of radius 2mm dropping from a height of 50mm on an aluminium plate of 500mm x 500mm x 2mm, with 4 piezoelectric sensors attached to the plate. These sensors record the wave propagation pattern in the plate after the impact. The goal of this study is to predict the coordinates of the impact of the ball using Neural Network models. To generate the input data to the model, numerical simulations for impact are performed (in the 1st quadrant) in ABAQUS. The final task is to predict the co-ordinates of unlabelled experimental data which will be referred to as "Test Data" further on. Since the input data is available only in the 1st quadrant, it is augmented in the other 3 quadrants by exchanging the sensor labels. After this, we obtain 925 impact locations[1].

Feed Forward Neural Network (FFNN), K-Nearest Neighbour Regression (KNN) and a Convolution Neural Network (CNN1) [2] were previously implemented using TensorFlow [3] and Keras [4] and compared based on their respective scores. In this study, the structure and parameters of CNN1 are improved by leveraging the geometric relations between the sensors. Two separate optimized networks are developed for predicting X and Y coordinates individually. The considered regression problem is also converted to a classification problem for analysing the network performance on different formulations.

Since the numerical data is close to ideal conditions, a 'controlled noise' was added to the numerical data so that it could mimic the experimental data. This artificial augmenting procedure generated a more robust CNN7.

In addition, Ensemble Learning Methods [2] are applied by using Random Forest (RF) architecture to develop a low memory requirement network for onboard applications. The application of generative algorithms using GANs and VAEs are developed and their applicability and limitations for considered task are discussed.

METHODS:

A. Original Models (KNN, FFNN, CNN1)

KNN model was the simplest model used, with the 4 nearest neighbours chosen. FFNN model is slightly more complex than KNN, with 5 hidden layers. CNN1 [5] is more complex than the previous two architectures but gave the best results. tanh is used as an activation function since

[1] These 925 datapoints are very close to ideal situations since they are obtained through numerical simulations. The experimental data points are far from ideal conditions. Hence, experimental validation data is included in the training for better predictions.

it gives equal probability of predicting positive and negative values. RMSE on the test data was 8.6 mm.

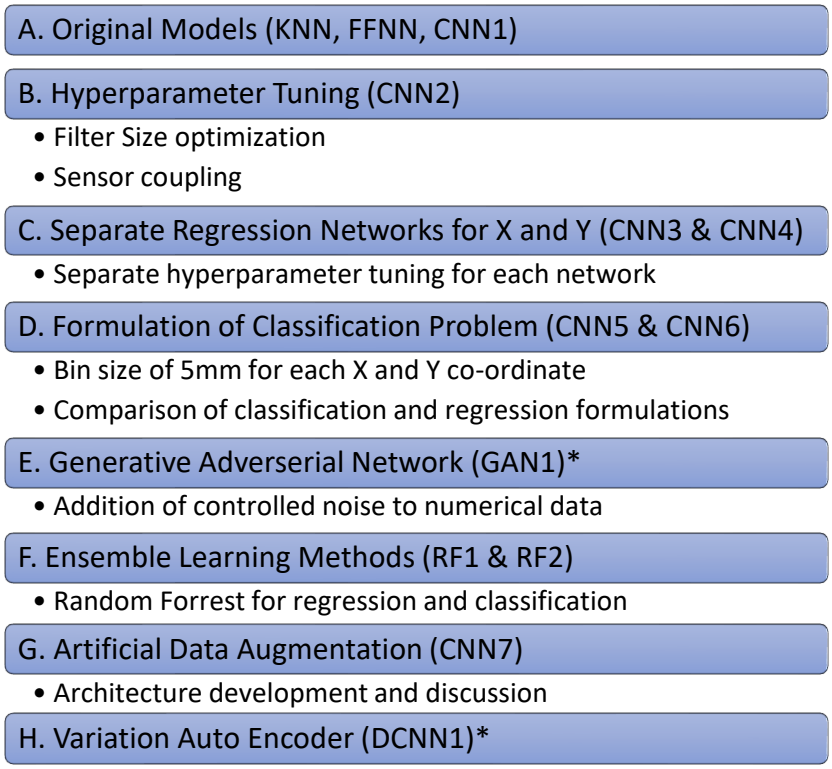

Figure 1: Workflow of the Report (*Not pursued further)

B. Hyperparameter Tuning (CNN2)

The following changes are done to CNN1 to optimize the network:

1. Hyperparameter optimisation of CNN1 using brute force search for number of filters, pooling size and filter size (70 iterations)
2. Output size of CNN1 was (4 x 4), where each column represented individual sensors. The output of CNN2 is (4 x 2), so that the network finds the corelations between sensors.
3. Iterating with different combinations of sensor coupling.
4. Augmenting data by reversing the calibration of sensors as the experimental sensors were calibrated in a negative manner than compared to that of numerical sensors

C. Separate Regression Networks for X and Y (CNN3 & CNN4)

Based on CNN2, two different regression CNNs for prediction of X (CNN3) and Y (CNN4) co-ordinate were developed *separately* (Figure 2) to understand if they need different networks for prediction. RMSE of X co-ordinate prediction was less than that of Y by order of 1e-2. Hence, hyperparameters were optimized for Y co-ordinate prediction separately.

CNN3 was also formulated as a classification problem. The optimised hyperparameters for each network were found by looping over a range of number of filters, pool size, filter size, initial learning rate and the learning decay rate which in total resulted in 200 iterations.

D. Formulation of Classification Problem (CNN5 & CNN6)

The motivation to convert regression formulation to classification formulation was based on measuring the performance of CNN3 and CNN4 on pre-defined bin size of 5mm and developing more sophisticated networks like Generative Adversarial Networks (GANs) for generating more training data points. The original coordinate axes ranging from -75 to 75 mm were divided into classes spaced at a uniform distance of 5 mm. The classes were one-hot encoded and represented by a (31 x 1) feature vector for both X and Y coordinates (Figure 2). The sensor values are normalized from 0 to 1 since the class label is always positive.

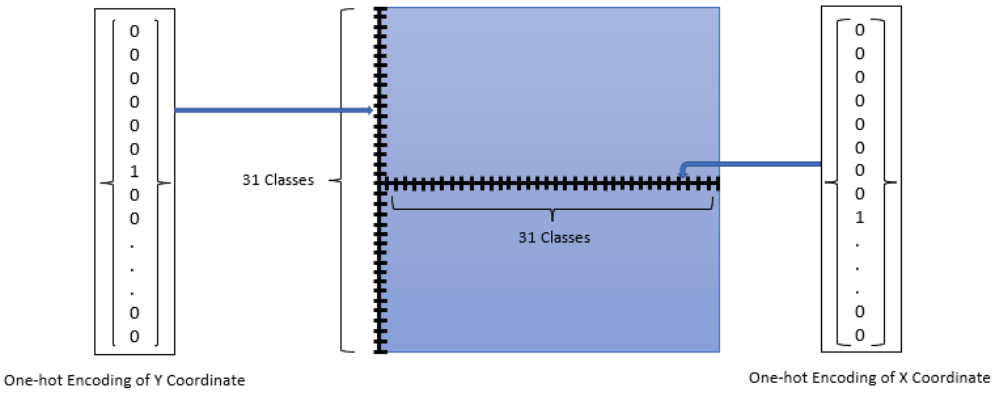

Figure 2: Conversion from regression formulation to classification formulation

E. Generative Adversarial Network (GAN1)

The motivation for further development of our models was to artificially generate data for the co-ordinates beyond the range given (-325 to 325 each axis), which once trained will be able to generate data very quickly compared to the Numerical Simulations and Experiments. Further idea was to understand how accurate the generated data is for a given setup of the sensors and to decide a range up to which extrapolation can be done without generating absurd data (due to normalization, the data will not be effective after a certain range).

The structure of GAN1 [6] for the classification problem was developed to create more training data for the CNN7. Different generators were developed for each class to generate more data for supervised learning. By adopting this technique data can be extrapolated to other points by changing the latent input vector. As the sensor time series data cannot be evaluated visually (for example like the MNIST data) it was not possible to assign a label to the data generated by controlling the values of the latent input vector. A code was developed for this approach but did not pursue this further due to the training time and a smaller number of available input training data for the discriminator network for a particular point.

F. Ensemble Learning Methods (RF1 & RF2)

During searching for the optimised hyperparameters, the bottleneck is the time consumed. Also due to memory requirements of the CNN network, it is difficult to incorporate it in an onboard application (Microcontrollers). Hence, a faster method using Ensemble training for Random Forest Regression (RF1) [2] and Classification (RF2). Ensemble methods make the use of an array of weak classifiers (Decision Trees) to obtain the result. The low evaluation time of this ensemble method is regarded to its structure. We split the current problem in X and Y coordinates and trained two different ensembles on them. Bootstrapped samples are used while building trees since they perform best. The structure hyperparameters of number of estimators and number of features were arrived at after performing brute force search (60 Iterations).

Table 1: Hyperparameters for RF1 and RF2

Hyperparameters	***RF1 (X-R)***	***RF1 (Y-R)***	***RF2 (X-C)***	***RF2 (Y-C)***
Experimental Validation Data	Yes	Yes	Yes	Yes
Controlled Noise	No	No	No	No
Estimators (Number of Trees)	1000	1000	500	500
Maximum Features	16	16	256	256

G. Artificial Data Augmentation – Controlled Noise Modelling (CNN7)

It was observed that the experimental data had a relative phase shift for each sensor. The motivation behind artificial augmentation was to generate more data mimicking the experimental data, since the experimental data is available only at a few selected points. By adding a controlled noise to the sensor data, we generated 42 times the original numerical data, which incorporated the phase-shifts of the experimental data. The 42 multiples are obtained by the following steps:

- Shifting 1 sensor data to right and left: 8 extra data points
- Shifting 2 sensor data simultaneously to right and left: 12 extra data points
- Shifting 3 sensor data simultaneously to right and left: 8 extra data points
- Inversing the amplitude of every sensor: Doubles the previously obtained points (2*29=58)

This done by adding a *controlled noise* to the sensor data (Figure 3). The numerical data is shifted by 2.5e-6 sec[2]. After this, the numerical and experimental data is cut for a window of 0.2ms, resampled to 500 points and locally normalized[3]. A wider architecture CNN5 is used to train this augmented data. Due to huge number of trained parameters and data sets, GPU[4] accelerated runs [7] were required for training. Optimization of hyperparameters was done trivially for some certain parameters, but the results are very promising, as it reduced the RMSE quite significantly. If experimental data could be modelled more accurately, then this data set for each point can indeed be used to train and model the generator and discriminator of the GAN.

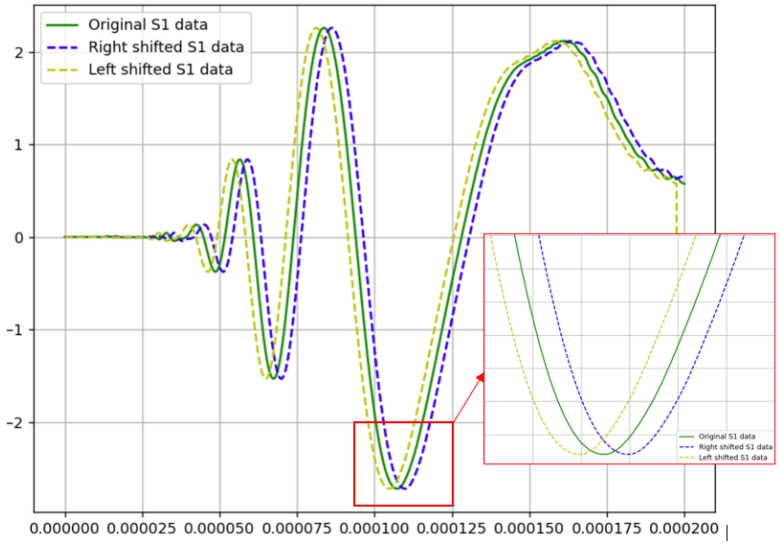

Figure 3: Effect of Adding Controlled Noise to Sensor Data

Table 2: Hyperparameters for FFNN, CNN1, CNN2, CNN3, CNN4, CNN5, CNN6 and CNN7

Hyperparameters	*FFNN*	*CNN1*	*CNN2 (R)*	*CNN3 (X-R)*	*CNN4 (Y-R)*	*CNN5 (X-C)*	*CNN6 (Y-C)*	*CNN7*
Experimental Validation Data	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Controlled Noise	No	No	No	No	No	No	No	Yes
Initial Learning Rate	0.0015	0.001	0.001	0.0015	0.001	0.00158	0.001	0.001
Decay Steps	100	1000	1000	1000	1000	1000	1000	5000
Decay Rate	0.9	0.7	0.7	0.75	0.7	0.753	0.7	0.7
Regularizer	L2(0.01)	N.A.	N.A.	N.A.	N.A.	L2(0.005)	L2(0.005)	N.A.
Batch Size	32	10	15	15	15	15	15	28
No of Filters	N.A.	(32, 64, 64)	(28, 56, 56)	(40, 80, 80)	(44, 88, 88)	(40, 80, 80)	(44, 88, 88)	(128, 256, 256)
Parameters	2184958	97714	81306	185665	224289	185551	224167	1677106
Filter Size	-	(5,1)	(5,1)	(7,1)	(7,1)	(7,1)	(7,1)	(5,1)
Strides	-	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)

RESULTS:

The results of all the iterations performed are summarised in Table 3, in the chronological order of development of the models. The R2 or F – score, respectively for a regression or classification model, is computed on the test data generated while training (Train – Val – Test Split). Validation

[2] The resolution of the experimental data. 1e-6 s corresponds to 100 rows in numerical recording.

[3] A MinMaxScalar approach is not chosen, as we want to preserve the relative amplitude of our sensor signals

[4] Total datapoints are 38850 with approximately 1.6million parameters. Hence, Tesla V100-SXM2-16GB, compute capability of 7.0 was used available at the CLAIX [6] Supercomputer by RWTH Aachen University

MSE/Accuracy is computed on the validation data set for the normalised co-ordinates. RMSE and Max Deviation on Test I is computed on the split test data during training whereas the Test II data is the unlabelled experimental data which had to be predicted (whose labels were given after Phase 1 [1] of the Project). The aim was to improve the network architecture to perform the best on the Test II dataset.

Table 3: Summary of Results

Results (R: Regression C: Classification)	***R2 / F Score***	***Validation MSE / Accuracy***	***RMSE Test I (mm)***	***Max. Deviation Test I (mm)***	***RMSE Test II (mm)***	***Max. Deviation Test II (mm)***
KNN (R)	0.9846	0.0098	-	-	17.53	38.75
FFNN (R)	0.8238	0.019	-	-	19.35	43.65
CNN1 (R)	0.9976	0.00082	2.093	8.754	8.651	22.696
CNN2 (R)	0.99864	0.0003	1.6213	5.91	4.1	8.44
CNN3 (X-R)	0.99961	0.00014	0.8809	2.8	3.13	7.1837
CNN4 (Y-R)	0.99887	0.00038	1.4466	5.539	4.75	8.8027
CNN5 (X-C)	0.9655	0.961	0.222	10	0.8164	5
CNN6 (Y-C)	0.9802	0.974	0.1856	10	1.5456	10
RF1 (X-R)	0.9986	-	1.046	2.5	5.449	9.199
RF1 (Y-R)	0.9962	-	1.31	2.26	8.502	20.009
RF2 (X-C)	1	-	0	0	13.12	35
RF2 (Y-C)	1	-	0	0	25.522	50
CNN7 (R)	0.99696	1.18E-05	0.26	7.07	4.938	10.452

By implementing sensor coupling and hyperparameter tuning in CNN1, a significant improvement is seen in the RMSE in CNN2. Comparison of CNN3 and CNN4 reveals the need of improvement on the complexity of the Y prediction model, as the X co-ordinate is being predicted accurately with an error lower than for CNN2. CNN5 and CNN6 perform well too, with the least MSE for both Test I and Test II datasets. Only 13 datasets for X and 21 datasets for Y out of total 2016 datasets were classified incorrectly.

RF1 (regression) performed better than RF2 (classification). An important thing to note here was that the Classifier and Regressor both performed well on the entire dataset but performed poorly on the prediction of the Test II data set.

Finally, after adding controlled noise and excluding the experimental data from the training dataset, a better result is achieved using CNN7. In Phase I of the project, the MSE on the Test I dataset was 10mm (not included in the Table 3). However, after adding the controlled noise, this error has reduced almost by an order of 5.

DISCUSSIONS:

Since CNN1 gave promising results, we pursued optimising CNN1 further rather than KNN and FFCN networks. In all the iterations, care was taken to ensure that the model does not overfit. Regularizers, Dropouts, Batch Normalization and reduction in training parameters was done wherever overfitting was observed.

The further optimisation of CNN5 and CNN6 was not performed. In order to go ahead with GANs we need multiple data for a certain class, hence efforts were focused in that direction.

As Ensemble method is a collection of weak classifiers/Regressors trained to have a minimum probability of just over 0.5, it fits well only during the initial levels of model development for our problem. However, this method is easy to code and requires much less memory to train.

CNN7 performance on predicting the experimental data should be further pursued as there is more scope in improving the MSE, by devising more better ways to model the noise. Ideally

artificially augmented data can be used for all the other variations of networks, to study its effect on them but this was not done here as it is very computationally expensive.

Finally, a Variation of the Autoencoder Network (DCNN1) was tried to develop to generate data beyond the given range of co-ordinates. Autoencoders are fully Convolutional Networks which was initially modelled in UNET [8] architecture for semantic segmentation. They have an encoder network (Convolution) which reduces the input to a feature vector and a decoder network (De-Convolution) with skip connections to regenerate the input shape of data with the required features. In our case, it was tried to use DCNN1 along with its weights as encoder network. Then we trained a MLP network which would generate the latent input vector to the decoder network. Then a decoder network is made to get back the input data. Some problems with inverting the model (decoder) were encountered. Hence, due to time constraint, DCNN1 was not pursued further.

CONCLUSION:

The artificial augmenting network CNN7 can be further improved after modelling the noise more accurately. This will generate more data for a particular class which can further be used to generate data for GANs or Autoencoder networks. This data can also be used to make classification networks CNN5 and CNN6 more robust. Data can also be generated for different energy levels which can be further used to evaluate the damage severity of the impacts. Reversing the calibration and exchanging the sensor sequence improved the results. This is essentially because the numerical (input) data started resembling the experimental data. Coupling of sensors by reducing the second dimension of output of conv layers improved the results as it incorporated the inherent geometrical corelation between the sensors. Transfer learning can be used for more data to come up with more complex network to reduce the time needed for training. GANs and VAEs have a lot of potential when used in combination of artificial data augmentation.

REFERENCES:

[1] Prof. Dr. -Ing. B. Markert, Dr. F. Bamer, Dr. sc. ETH Dipl.-Ing. Georg Kocur & D. Thaler (M.Sc.), Computational Intelligence in Engineering Lecture Notes, Institut für Allgemeine Mechanik, RWTH Aachen University, 2021-2022.

[2] Prof. Dr. B. Leibe, Machine Learning Lecture Notes, Visual Computing Institute, RWTH Aachen University, 2020-2021.

[3] M. Abadi and others, “TensorFlow: A system for large-scale machine learning,” 2016. [Online]. Available: https://www.tensorflow.org/.

[4] F. Chollet & others, “Keras,” 2015. [Online]. Available: https://keras.io/.

[5] T., Iuliana; Fu., Hailing; Z. & Sharif Khodaei, “A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures,” Sensors, vol. 19, no. 22, p. 25, 2019.

[6] Prof. Dr. -Ing. M. Stoffel, R Gulakala (M.Sc.), “Artificial Neural Networks in Structural Mechanics Course,” 2022.

[7] RWTH Compute Cluster, “CLAIX,” RWTH Aachen University.

[8] “Towards Data Science,” [Online]. Available: https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5.