METHODOLOGY

The dataset used for training contained of 244 different hit measurements with spherical objects with constant mass on generated in numerical software plane. Hits were equally distributed over plane in 5mm interval over the y and x axis. Each measurement contained readings from four distributed on the plane piezoelectric sensors where each generated change in voltage over time during excitation. Further this data is called “numerical data” or “training data”. The dataset used for testing contained of 18 hit measurements on real plate performed in laboratory conditions. Hits were performed with spherical objects with different masses. Moreover, hits were distributed unequally. Further this data is called “test data” or “ experimental data”. The dataset used for validation contained of 24 hit measurements on real plate at the same conditions as the test data. Further this data is called “validation data” or “ experimental data”. To equalize sampling rate of the numerical(20000 points?) and experimental (points?) data every 100th measurement point of numerical data was considered. For each measurement normalization was performed based on maximal sensor value on both datasets. Experimental and numerical data was investigated and compared. A the end gaussian filter was applied with standard deviation equals 3.

The next stage was building artificial neural network classification models in Python using Keras API to predict hit coordinates based on preprocessed data. First architecture was an Feed forward model with one dense layer of 128 nodes activated by TANH activation function and output layer with softmax activation function. Second architecture was feed forward model with Convolution layers at the beginning. Convolution section was consisted of four main parts: four 1D convolution layers with Max pooling filtering the signals from each sensors (Kernels: 8, Kernel size: 16, Pooling: 8), concentration point where results were merged, expanding dimensions, 1D convolution layer with Max pooling at the end for merged results (Kernels: 8, Kernel size: 16, Pooling: 8). The merged results from the convolution section were passed to feed forward section consisted of following parts respectively: dense layer with 124 neurons and Relu activation function, Dropout (0.2), dense layer with 124 neurons and Relu activation function, output layer with softmax activation function. Third and the last architecture was feed forward model with Convolution layers at the beginning. The convolution section consisted of three layers with the same parameters; 1D convolution layer with Max Pooling (Kernels: 32, Kernel size: 8, Pooling: 2) and Dropout (0.05) between the convolution and pooling. Feed forward section consisted of following parts respectively: dense layer with 63 neurons and Relu activation function, Dropout (0.1), dense layer with 64 neurons and Relu activation function, output layer with softmax activation function.

Performance of all presented models was assessed by the loss function and a custom accuracy. The custom accuracy metrics was comparing predicted coordinates with correct ones and returns truth if the predicted position was within 5 mm range of the correct position. Total error was calculated by taking number of measurements fulfilling the requirement and dividing it over total test samples. To visually inspect probabilities custom heat map was developed pointing out probabilities in a color scale, predicted hit coordinates and correct hit coordinates.

All models were tested and through tests their parameters were changed: dropout, batch size, learning rate, kernel size, number of kernels, pooling size, activation functions, number of neurons and so one. Since tuning the neural network is not the topic of this paper, tuning methodology is skipped and the best performing models are presented above.

The third model gave the best results and it was chosen for further development. The first enhancement was function interpolating through output probabilities. (HOW IT WORKS??). The second one was modifying the input parameters by taking first and second derivatives of the signals as well as furrier transformation. Last step was testing the model with different combinations of input datasets and validating results using heatmap and custom accuracy.

RESULTS

DISCUSSION

The results of the study revealed that in this case implementing derivatives of the signals and furrier transformation doesn’t improve the model significantly or even decrease accuracy. The problem is connected with differences between experimental data used to validate and test, and numerical data used to train the model. Some of this differences (WHAT DIFFERENCES?) were successfully overcome by normalization, resampling and gaussian filter but the difference in signal characteristics of hits with different parameters was too big for model trained on regular numerical data signals. Since the convolution creates and image of the signal and by

Interpolation function significantly decreased the error in cases where tested signal was unnormal.